ESCUELA TÉCNICA SUPERIOR DE INGENIEROS INDUSTRIALES Y DE TELECOMUNICACIÓN

INGENIERÍA INFORMÁTICA

PROYECTO FIN DE CARRERA

"Diseño y desarrollo del módulo de comunicación e interacción de un e-Reader accesible"

Alumna: Maitane Itoiz Lleida
Tutor: José Javier Astráin Escola
Iruñea – Pamplona, 27 de junio de 2013

Índice

Lista de figuras

Capítulo 1

Introducción

En el año 2011, las empresas Navarras ***Job Accommodation*** y ***Leer-e*** se unieron en un proyecto empresarial colaborativo que tiene como objeto la realización de un e-Reader adaptado a personas con diferentes tipos de discapacidad funcional y de movilidad.

El objetivo de esta unión es complementarse mutuamente para conseguir realizar y comercializar el lector electrónico, ya que por un lado la empresa Job Accommodation, empresa que basa los desarrollos tecnológicos en ser un agente para personas con discapacidad, entre otras formas, diseñando o modificando productos para la facilitación y mejora de sus actividades de ocio o trabajo, se centraría en el desarrollo del producto. Y por otro lado Leer-e, empresa dedicada a ofrecer al mercado productos innovadores relacionados con la lectura digital, ya sea en cuanto a dispositivos o a contenidos digitales, se dedicaría a la expansión y comercialización del mismo.

El proyecto comenzó a desarrollarse a mediados de ese mismo año comenzando por analizar las posibles soluciones tecnológicas para poder llevar a cabo el proyecto. Cuándo se dio con la solución, entre otras surgió la necesidad de preparar la plataforma y la aplicación que manejaría todo el sistema. Durante el año 2011-2012 se llevó a cabo dicha tarea, realizando un análisis detallado del software a desarrollar, el diseño y su posterior desarrollo sobre la plataforma elegida.

La memoria se organiza en tres partes. Un breve análisis de la viabilidad del proyecto, una descripción del estado en el que se encontraba dicho proyecto cuando empecé a formar parte de él y por otro lado, los objetivos globales de este proyecto fin de carrera y el desarrollo realizado para la consecución de los mismos.

1.1 Antecedentes y descripción del problema

Los modelos de lectores electrónicos existentes en el mercado integran gran parte de las características de accesibilidad orientadas a usuarios con déficit de visión. Sin embargo, hasta ahora no se han tenido en cuenta a aquellos usuarios que tienen movilidad reducida o problemas de coordinación y para los cuales, un e-Reader convencional resulta inaccesible, puesto que para muchos de ellos es imposible manejar los botones integrados en estos aparatos y que realizan funciones tan simples como encenderlo o apagarlo.

Según un estudio realizado por la empresa Job Accommodation, este es el listado de características a tener en cuenta en la accesibilidad de los e-Reader.

Características obligatorias

Si un usuario no puede llevar a cabo las siguientes funciones básicas, un e-Reader no es accesible:

- Escuchar la versión audio de un texto electrónico.
- Manejar los controles básicos como subir/bajar el volumen.
- Abrir un texto y navegar por él (avance/retroceso de páginas).
- Navegar por el menú de la aplicación

Características recomendadas

Para manejar el e-Reader:

- Función 'text-to-speech' para todos los textos digitales.
- Señal audible o táctil de las funciones de control (on/off, volumen).
- Señal audible de las funciones operacionales (selección de libro, anotaciones, búsqueda, diccionario).
- Botón físico para el contraste de la pantalla y para aumentar/reducir el tamaño de letra.
- Entrada para un teclado u otro dispositivo que ayude al usuario a utilizar el e-Reader.
- Entrada para dispositivos de usuarios con movilidad reducida, como controles con la boca.

Para navegar por los libros electrónicos:

- Señal audible que anuncie la posición actual (libro, capítulo, página).
- Información audible sobre los marcadores de página, anotaciones y datos similares existentes en la vista actual.

1.2 Objetivos

Como se ha comentado anteriormente, la mayoría de lectores electrónicos del mercado integran gran parte de las características de accesibilidad orientadas a usuarios con baja visión.

La idea principal de este proyecto es preparar el software de una placa de desarrollo para que pueda servir como plataforma para un e-Reader accesible. Además, también se deberá crear una aplicación de lectura de libros electrónicos para la plataforma, y cualquier tipo de tablet.

El diseño del e-Reader accesible se basa en un e-Reader convencional que además, aporta las siguientes características:

Interfaz gráfica amigable y usable

El e-Reader accesible debe ser lo más usable posible, es decir, lo más fácil y claro posible a la hora de usar, empezando por su interfaz gráfica.

Controles accesibles

Los botones que se incluyan en la plataforma deberán estar preparados para facilitar el uso del e-Reader. Para eso deberán ser mayores de lo normal, con inscripciones, de manera que se puedan reconocer al tacto y deberán tener una colocación y separación adecuada para facilitar su uso.

Control remoto y sin cables

El usuario utiliza el e-Reader a través de un dispositivo interfaz adaptado a él en lugar de emplear los botones y/o pantalla táctil incluidos en el e-Reader. El control del e-Reader se lleva a cabo sin cables y mediante cuatro botones y un joystick, lo cual repercute positivamente en la usabilidad del producto y en la comodidad y facilidad de uso.

1.3 Estado del arte

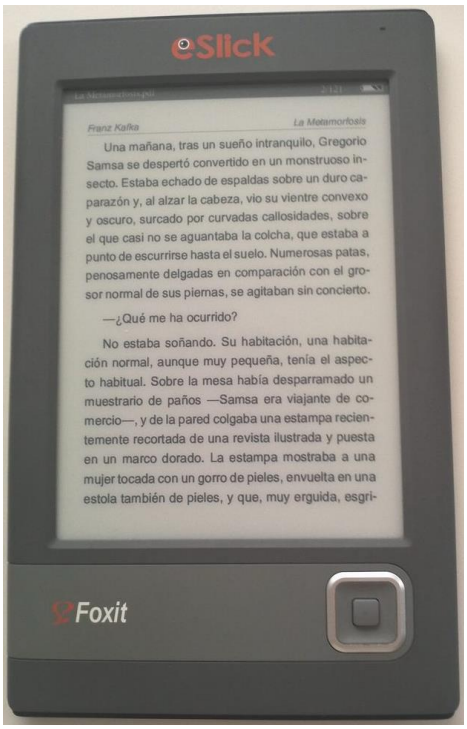


Figura 1: Lector eSlick de Foxit

En este apartado se toma como referencia un modelo concreto de e-Reader ampliamente extendido, el modelo eSlick de Foxit, para conocer lo que se ofrece actualmente en el mercado así como sus limitaciones hacia determinados usuarios.

En la figura 1 se muestra una vista completa del lector que tiene las siguientes características:

- Pantalla: 6 pulgadas
- Tinta electrónica
- Dimensiones: 188x118x9.2mm
- Peso: 180gr

Tal y como se puede comprobar, las limitaciones de este lector, al igual que la mayoría de los actuales, se pueden diferenciar claramente en dos grupos: aquellas relativas a la interfaz gráfica y las de los botones del dispositivo. En esta parte del proyecto nos hemos centrado en lo correspondiente al manejo de los botones.


Figura 2: Botones laterales del lector eSlick de Foxit

En la figura 2 se muestra una vista más detallada de los botones de navegación que podemos encontrar en el dispositivo. Queda notablemente visible que los botones que sirven para acceder a los menús principales así como los de navegación no son nada accesibles para usuarios que tengan cierta movilidad reducida. Vamos a hacer un pequeño estudio más detallado de cada uno de los botones:

- La disposición de los botones en el lateral del e-Reader hace que el acceso a ellos sea complicado.
- Su tamaño, al igual que el espaciado entre los mismos, es muy reducido. Se podría pulsar más de un botón al mismo tiempo.
- Podemos ver que apenas sobresalen de la carcasa, lo que hace que su localización sea difícil.
- Los iconos que representan su utilidad son poco identificables a la vez que muy pequeños. Los dibujos no contrastan con el fondo del botón.

Por otra parte, en la parte frontal del dispositivo, se encuentra un botón que permite la navegación por los menús y por los libros electrónicos. Podemos ver en la figura 3 cómo este botón es poco usable ya que con un mismo botón se pueden realizar acciones muy diferentes. Es decir, el marco plateado sirve para moverse por las direcciones arriba, abajo, izquierda y derecha y el hecho de que todo ello se haga con un solo botón (y no cuatro independientes) dificulta su manejo.

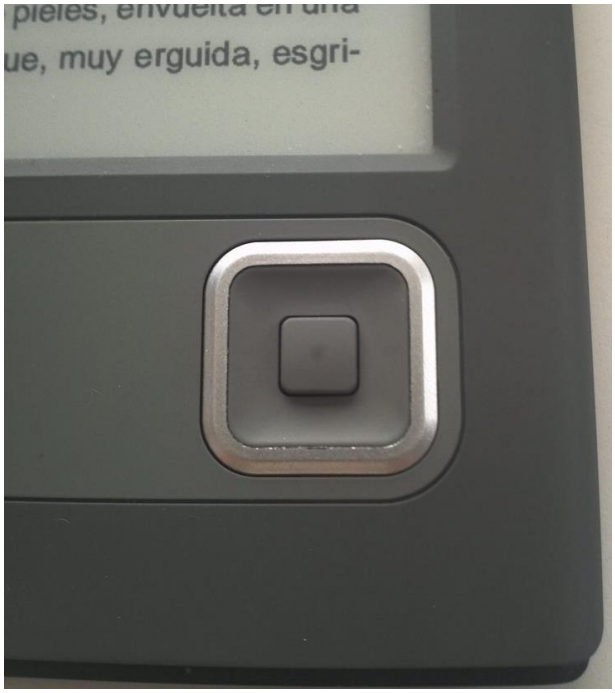


Figura 3: Controlador eSlick de Foxit

1.4 Solución propuesta

El e-Reader que se desea realizar deberá ser diferente de cualquier lector electrónico convencional, de manera que permita que usuarios que presenten algún tipo de discapacidad motora puedan disfrutar de la lectura de libros digitales, que de otra manera, no sería posible.

Según el esquema de la figura 4, a través de un dispositivo externo (*botonera*, joystick, pedal, soplador, etc.) el usuario puede manejar el e-Reader exactamente igual que si lo hiciera a través de los botones integrados en el dispositivo. Por ejemplo, se supone un dispositivo externo que dispone de un joystick que puede moverse en las direcciones arriba, abajo, izquierda, derecha. De este modo, al mover el joystick en la dirección abajo, el dispositivo externo envía un mensaje al e-Reader para que se mueva en el menú principal del icono actual, más concretamente hacia abajo. Hará lo correspondiente con las direcciones de arriba, izquierda y derecha. También cuenta con tres botones que permitirán simular las acciones de intro, salir y menú.

Figura 4: Esquema de interacción hombre-máquina

Para el diseño de este dispositivo se hizo un pequeño brainstorming con varios miembros de la empresa pidiéndole consejo, concretamente, a uno de los miembros que tiene cierta limitación visual. Finalmente, el prototipo que se consiguió diseñar fue un controlador inalámbrico y como ya he comentado más arriba, consta de un joystick y tres botones. Además de proporcionar comodidad al usuario, de esta forma se evita que sea el usuario el que se adapte al producto. El usuario puede colocar y utilizar el control remoto como mejor le convenga y manteniendo en todo momento la comunicación con el e-Reader.

1.5 Procedimiento para la solución

Antes de explicar cuál ha sido el proceso y metodología que se han desarrollado en el tiempo que ha durado el proyecto, se van a detallar las fases por las que se ha pasado y deberá pasar el proyecto en su totalidad.

1.5.1 Análisis

- ***Definición del sistema***

El primer paso consiste en definir el tipo de sistema que se quiere realizar, qué prestaciones debe tener, funcionalidades, etc. Quizá este es uno de los más importantes ya que en caso de no concretar correctamente las características del producto habrá que hacer modificaciones más adelante y esto puede tener consecuencias graves en el desarrollo del producto.

- ***Selección del hardware***

Una vez establecidos los requisitos, se empiezan a definir los elementos específicos a utilizar. Esta decisión deberá tomarse conjuntamente con el siguiente punto, la selección del software, ya que ambos son dependientes uno del otro. Para la selección del hardware habrá que tomar las siguientes decisiones:

- *Evaluar la tecnología inalámbrica:* Se deberá hacer un análisis de las posibles soluciones que puedan cumplir con los requisitos establecidos en el proyecto.
- *Comprobación de la compatibilidad del sistema operativo con la tecnología inalámbrica:* Para poder tomar esta decisión, habrá que tener en cuenta tanto la plataforma utilizada como el sistema operativo utilizado al desarrollar la aplicación.

- ***Selección del software***

De la misma manera que se ha seleccionado el hardware a utilizar, se deberá decidir cómo se manejará el módulo periférico para controlar la aplicación.

1.5.2 Diseño

Definidos ya los requisitos que debe cumplir el sistema, definiremos los elementos específicos que vamos a utilizar. Se pasará por las siguientes fases:

- ***Diseño del dispositivo inalámbrico***

En esta fase, deberá diseñarse el dispositivo de manera que sirva al usuario como control remoto, es decir, como *botonera*, joystick, pedal, soplador o una combinación de ellos. Una vez decidido el dispositivo concreto a utilizar, y en caso de que se haga una combinación de varios dispositivos, habrá que definir los elementos que va a tener.

1.5.3 Desarrollo

Utilizando toda la información recopilada hasta el momento, se comenzará con el desarrollo tanto de la parte de hardware como las modificaciones necesarias que haya que hacer en la parte de software.

1.5.4 Pruebas

Una vez desarrollado todo el sistema, habrá que diseñar y aplicar una serie de pruebas para comprobar que se han cumplido todos los requisitos establecidos y que todo lo desarrollado funciona correctamente.

Estas pruebas serán modulares como de conjunto, asegurando de este modo, el correcto funcionamiento de todo el sistema.

1.5.5 Mantenimiento

Esta se consideraría la fase final de la creación del producto, aunque en realidad esta fase podría alargarse a lo largo del tiempo ya que una vez comercializado el producto habrá que darle soporte al usuario además de arreglar los posibles fallos que aparezcan e ir aplicando las mejoras necesarias conforme la tecnología avance (como por ejemplo, la aparición de nuevas versiones del sistema operativo) para que siga siendo competitivo en el mercado.

1.6 Estado previo del PFC

Después de haber explicado de qué trata el proyecto y cuáles son las etapas que se van a desarrollar, a continuación se analizará en qué fase se encontraba el proyecto cuando yo empecé a formar parte de él y qué decisiones fueron tomadas antes de mi llegada, ya que algunas de estas elecciones han marcado el desarrollo del mismo.

A finales de agosto llegué a Job Accommodation y para entonces toda la parte de creación de un nuevo software que mejorase los actuales e-Readers ya estaba desarrollado. Los objetivos y la solución a los problemas comentados anteriormente ya estaban resueltos y quedaba ponerse a desarrollar la parte de hardware. Al tener los requisitos y lo que se quería realizar bien definido, se había preparado también un boceto de lo que sería la *botonera* a implementar pero antes de seleccionar una solución de los diferentes módulos que se iban a utilizar, se realizó un estudio de cuáles podían ser las diferentes alternativas y cuál sería la que mejor satisficiese las necesidades definidas.

Capítulo 2

Análisis

2.1 Objetivos generales del sistema

Tal y como se ha especificado en la introducción, se desea crear un dispositivo periférico que permita el manejo de un e-Reader que esté adaptado a personas con diferentes tipos de discapacidad funcional y de movilidad reducida.

Se podría considerar que ya hay muchos modelos distintos de lectores de libros electrónicos en el mercado y que competir contra ellos es una batalla perdida pero desde Job Accommodation se consideró que dichos dispositivos no son accesibles a todo tipo de público ya que hay un sector de la población que no puede hacer uso de ellos. Por ello, se añadirán una serie de opciones específicas para poder hacer este nuevo producto accesible sin perder ninguna de las ventajas que esta tecnología ofrece.

Se deberá realizar una conexión virtual entre el dispositivo externo y el e-Reader de modo que no haya ningún cable que dificulte su usabilidad. Además, se crearán botones más adaptables, así como sus funcionalidades, a los diferentes tipos de necesidades de los usuarios.

Al final, lo que se pretende con este producto es satisfacer una necesidad que no está cubierta por la tecnología actual.

2.2 Definición de requisitos del sistema

Lo primero de todo, se decidió que se iba a dejar de lado la placa de evaluación que se estaba utilizando hasta entonces y se iba a migrar todo el sistema a Tablets. La migración era sencilla ya que sobre la placa de evaluación estaba corriendo el mismo sistema operativo que tendría la nueva Tablet que se iba a utilizar. El único inconveniente se encontraba en la versión que estaba corriendo sobre la Tablet y la versión que necesitaba el módulo bluetooth para trabajar. El perfil HID del módulo bluetooth que se iba a utilizar requería que la

versión de Android fuese superior a 3.1 y actualmente se estaba trabajando sobre la 2.3. Por ello, se debería realizar una actualización del sistema a una versión superior.

En cuanto a la parte de software de la aplicación, se deberá modificar el código ya desarrollado para poder acoplar las nuevas funcionalidades del dispositivo externo.

El propio sistema operativo del dispositivo (Android en este caso) ofrece una opción de habilitar/deshabilitar el bluetooth además de buscar los dispositivos que estén dentro del alcance pero acceder a este menú es complejo ya que hay que navegar por diferentes submenús. Por ello, se decidió crear una nueva clase que fuese sencilla y que permitiese realizar las mismas funciones. Esta nueva clase estará integrada dentro de la aplicación ya creada y constará de dos botones. Uno de ellos permitirá al usuario activar el bluetooth mientras que el otro lo que hará será desactivarlo. Además, una vez activado, aparecerá un botón (hasta ahora oculto) que permitirá al usuario buscar los dispositivos que estén dentro del alcance de nuestro módulo y los irá mostrando en una lista.

Por otra parte, estará el diseño del dispositivo externo en sí. Sabiendo a qué tipos de usuarios está orientado el producto, es necesario que la navegación por el sistema sea lo más sencilla posible.

Se requiere de tres botones que simulen las principales acciones que un usuario realiza sobre la aplicación. Dichas opciones serán *Intro* para acceder a las diferentes opciones, *Salir* para poder navegar hacia atrás entre las pantallas y *Menú* para poder acceder a los submenús que ofrece la aplicación. Haciendo un pequeño análisis de cómo deberían colocarse los botones, se llegó a la decisión de que lo más intuitivo era diseñar la botonera de modo que los botones se encontrasen en la parte derecha y que estuviesen en el siguiente orden:

- Botón de la derecha: Intro
- Botón de la izquierda: Salir
- Botón de arriba: Menú

En cuanto al manejo de las direcciones arriba, abajo, izquierda y derecha se decidió que se realizaría mediante un joystick ya que analizando los posibles usuarios potenciales de la aplicación, se consideró que era el método más sencillo para llegar a todo tipo de público.

Estos elementos irían soldados a una placa, la cual iría conectada a otras dos placas (una en la que recaería la programación de los botones y otra para realizar la conexión inalámbrica).

Para realizar la conexión inalámbrica, se consideraron dos opciones: wifi y bluetooth. Después de analizar las dos posibles alternativas, se decidió implementar una conexión bluetooth entre la botonera y el dispositivo ya que la

mayoría de dispositivos actuales disponen de bluetooth o sino, su adaptación es más sencilla que la del protocolo wifi. Es decir, existen diversos adaptadores que permiten tener conexión bluetooth mientras que realizar esa adaptación con un módulo wifi es más complejo.

Definidos ya dichos requisitos, el último paso era decidir qué placa base se utilizaría para que fuese la encargada de la programación de las acciones a realizar. La decisión de qué tipo de placa utilizar estaba bastante cerrada al definir los dos criterios anteriores, por lo que se pensó que los más sencillo era seguir por dicho camino y utilizar un módulo de Arduino.

2.2.1 Requisitos de hardware

2.2.1.1 Proceso de la selección de los diferentes módulos de la botonera

Tal y como se ha comentado anteriormente, desde un primer momento se decidió que la *botonera* iba a constar de tres partes: una placa de Arduino en la que recaería la programación de los mensajes a enviar al dispositivo, un módulo bluetooth para hacer la conexión remota y la propia *botonera* en sí, la cual tendría dos partes: un joystick para manejarnos por el menú y realizar las acciones de avance/retroceso de páginas y tres botones que simularían las funciones básicas de los botones que tienen los e-Reader (acción de intro, salir y menú).

La decisión del tipo de placa de Arduino (Arduino UNO en este caso) y de los botones propios de la *botonera* estaba tomada pero quedaba decidir qué tipo de módulo bluetooth se iba a utilizar ya que dependiendo de lo elegido, el desarrollo del envío de mensajes variaría mucho. También había que buscar una placa que contuviese el joystick y los tres botones que necesitábamos para crear nuestro prototipo.

Se plantearon dos alternativas para el bluetooth, ambas de la marca *SparkFun Electronics*:

- ***Bluetooth modem – BlueSMiRF Gold***

Se trata de la última sustitución de cable serie inalámbrica Bluetooth de SparkFun Electronics. Estos módems funcionan como una pipe de serie (RX/TX), es decir, cualquier flujo de datos en serie de 2400 a 115200bps se puede pasar sin problemas desde el ordenador al destino. El alcance máximo es de 106m y acepta alimentación de 3,3V hasta 6V. En caso de necesitar conectarlo al ordenador, se necesitará diseñar un circuito conversor de RS232 a TTL. Sus dimensiones son de 42x16.5x5.6mm.

- ***Bluetooth modem – BlueSMiRF HID***

La diferencia con el módulo anterior es que este viene cargado con el perfil HID. Este perfil, también llamado "dispositivo de interfaz humana", es el protocolo de comunicación utilizado para periféricos tales como teclados, ratones y joysticks. Esto hace que la RN-42-HID sea una herramienta sencilla y potente para la creación de dispositivos periféricos inalámbricos que pueden ser universalmente reconocidos y utilizados sin necesidad de instalar controladores especiales. La unidad de control remoto se puede alimentar de 3.3V a 6V. Se necesitará un circuito convertidor RS232 a TTL si se quiere conectar a un ordenador.

2.2.1.2 Alternativa seleccionada y justificación

- ***Módulo Bluetooth***

En un principio, se empezó a trabajar con el módulo ***BlueSMiRF Gold*** ya que contábamos con un ejemplar en la empresa. Pero tras varios análisis (que se detallarán más adelante) se optó por cambiar a ***BlueSMiRF HID*** ya que este módulo facilitaba mucho el trabajo a realizar recayendo todo el peso de la programación sobre la placa de Arduino.

El módulo bluetooth *BlueSMiRF Gold* envía el mensaje que se le pasa desde la placa de Arduino directamente al dispositivo, como una cadena de texto plano. En este caso, la aplicación que nosotros hemos programado es la encargada de recoger dicho mensaje y tratarlo para que haga las funciones que correspondan. El problema de este módulo era que el mensaje era enviado mediante un pipe desde la placa de Arduino (y mediante conexión bluetooth) al ordenador. A la hora de recibir el mensaje en la aplicación, convertirlo a evento del sistema era una tarea ardua ya que cada clase de la aplicación tendría que tratar todos los posibles eventos que le llegasen. Esto obligaba a tener varios hilos corriendo continuamente para tratar los mensajes que se fuesen recibiendo y surgía la posibilidad de que el procesador del dispositivo en el que estaba corriendo la aplicación no fuese lo suficientemente potente y la aplicación se

cerrase. Por ello, se pasó a buscar otras posibles alternativas hasta que se dio con la opción de utilizar un módulo bluetooth con perfil HID.

Como ya se ha comentado más arriba, al elegir este tipo de perfil toda la programación recae sobre la placa de Arduino. Este módulo convierte directamente los mensajes que le llegan desde la placa en eventos del sistema, por lo que evita todo el tema de tratamiento de mensajes en la parte receptora. Al pulsar la tecla de salir, el dispositivo receptor recibirá la señal de salir y realizará la acción correspondiente. Además, de este modo, se conseguía desarrollar un dispositivo que fuese genérico, es decir, que funcionase en cualquier dispositivo que tuviese conexión bluetooth sin limitarlo exclusivamente a la aplicación que nosotros habíamos desarrollado.

Características técnicas del módulo

- Incluye firmware configurado para el protocolo HID
- Tarjeta de desarrollo con interfaz UART
- Entradas/salidas con lógica de 3.3V o 5V
- Modulo Bluetooth calificado 2.1/2.0/1.2/1.1
- Soporta Bluetooth v2.1+EDR
- UART soporta ratas de baudios de 1200 a 3Mbit
- Soporta ratas de datos SPP - 240Kbps (slave), 300Kbps (master)
- Soporta ratas de datos HCI - 1.5Mbps, 3.0Mbps
- Soporta protocolos de interface BlueTooth SPP/DUN y HCI
- Dispone de software para modo HCI ó SPP/DUN
- Alcance: hasta 20m con línea de vista.
- Frecuencia: 2.402 ~ 2.48 GHz
- Modulación: FHSS/GFSK (79 canales a intervalos de 1MHz)
- Comunicación segura, encriptación de 128 bits
- Corrección de errores
- Potencia de salida: 4dBm
- Sensitividad: -80dBm
- Rata de transmisión no estándar programable: 1200bps hasta 921Kbps
- Consumo de corriente en transmisión: 60mA
- Consumo de corriente en recepción: 35mA
- Voltaje de alimentación: 3.3V (5 a 16V en header B)
- Tamaño: 15mm x 43mm
- Referencia SparkFun (WRL-10938)

- ***Placa de Arduino***

Figura 5: Placa Arduino UNO

En cuanto a la base de todo, se decidió que se tomaría una placa del tipo Arduino UNO. Se trata de una placa electrónica basada en el microprocesador Atmega328. Tiene 14 pines digitales de entrada/salida, 6 entradas analógicas, una conexión USB, conector de alimentación, cabecera ICSP y un botón de reinicio. Contiene todo lo necesario para apoyar al microcontrolador, basta con conectarlo a un ordenador con un cable USB. Además, utiliza un entorno de desarrollo muy fácil y al ser un producto Open-Source, cuenta con una amplia comunidad detrás.

Especificaciones técnicas:

- Microcontrolador: ATmega328
- Voltaje de entrada (recomendado): 7-12V
- Voltaje de entrada (límites): de 6-20V
- Pines Digital de entrada/salida: 14 (de los cuales 6 proporcionan salida PWM)
- Pines de entrada analógica: 6
- Memoria Flash: 32 KB (ATmega328) de los cuales 0,5 KB son utilizado por el gestor de arranque
- Velocidad de reloj: 16 MHz

- ***Botonera***

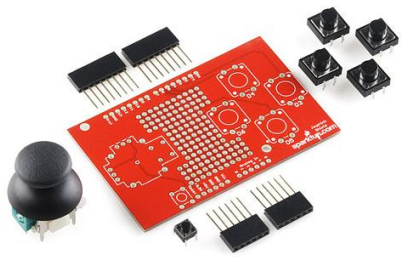


Figura 6: Kit joystick Shield

Para la *botonera* se decidió comprar un *Joystick Shield kit* de la misma marca que el bluetooth, *SparkFun Electronics*, que contenía todas aquellas partes que nosotros precisábamos. La placa que contiene los botones se acopla a la parte superior de la placa de Arduino y lo convierte en un simple controlador. Contendrá cinco pulsadores momentáneos (4 botones propiamente dichos más el joystick al presionarlo hacia abajo) y un joystick pulgar de dos ejes. Se requería de soldadura pero era relativamente sencillo su acople. Además, los pulsadores están conectados a los pines digitales de Arduino directamente lo que hace que la programación sea mucho más fácil. El movimiento vertical de la palanca de mando producirá un voltaje analógico proporcional en el pin analógico 0, del mismo modo, el movimiento horizontal de la palanca de mando se puede seguir en el pin analógico 1.

Una vez definida la estructura del hardware, se pasó a debatir cuál sería la funcionalidad de cada elemento de la *botonera* quedando del siguiente modo:

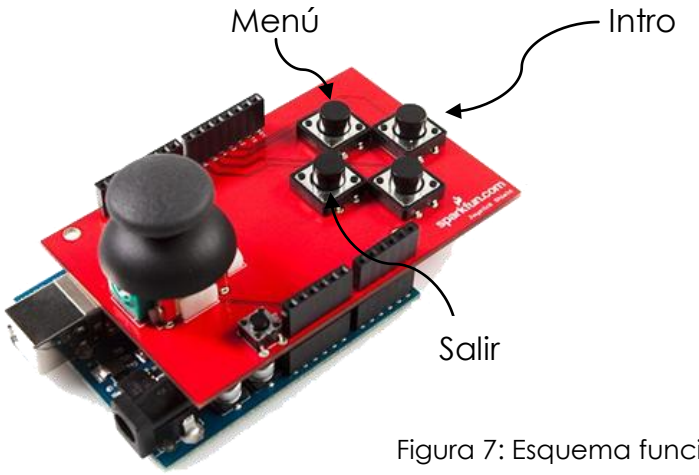


Figura 7: Esquema funcional de la botonera

2.2.2 *Requisitos de software*

En cuanto a la selección del software, poco había que decidir cuando yo empecé a formar parte del proyecto.

El sistema operativo en el que está basado el e-Reader estaba decidido. Cuando se empezó con este proyecto, se implementó sobre una placa IMX 508 EVK de Freescale la cual permitía como sistema base Android o un Ubuntu para sistemas embebidos. En un principio Linux era la opción que iba a ser seleccionada ya que todos los e-Readers que se habían analizado usaban este sistema además de ser el sistema operativo que venía de fábrica con la placa. Sin embargo, una vez puestos a analizar las posibilidades que ofrecía Android, no hubo lugar a duda entre una plataforma y la otra. Además, pensando en una posible migración a tablets, Android facilitaba muchísimo el camino.

Por un lado, Android incluye su lenguaje propio de desarrollo con su SDK y toda una amplia comunidad de desarrolladores y documentación. Se trata de una plataforma orientada a telefonía y tablets y por lo tanto su programación ofrece una amplia serie de funciones de capturas de eventos así como la opción de crear interfaces mucho más atractivas y simples de manera rápida y sencilla.

Tal y como se ha comentado al inicio de este capítulo, cuando se estaba buscando el módulo bluetooth a utilizar, se tuvo que tener en cuenta la versión de Android que se estaba utilizando ya que los perfiles HID sólo funcionan con aquellas versiones de Android superiores a la 3.1. Por ello, antes de comenzar a programar nada se tuvo que actualizar la versión de Android que tenía la Tablet ya que por aquel entonces se estaba trabajando sobre la 2.3. Finalmente, se

decidió actualizar a la última versión disponible en el mercado, la *Ice Cream* en su versión 4.0.

Por otro lado, para realizar las correspondientes pruebas con la placa de Arduino y el módulo bluetooth y comprobar que efectivamente se mandan los mensajes que se pretenden, se utilizará la aplicación zTerm para capturar los mensajes que se están enviando por los puertos.

- ***Lenguajes de programación***

Como la aplicación corre sobre Android, el lenguaje a utilizar es el definido por la propia plataforma.

Por otro lado, para realizar la programación de la placa de Arduino, se utilizará el lenguaje propio de Arduino en su entorno de desarrollo.

2.2.3 Análisis técnico

- ***Rendimiento***

El e-Reader debe responder ante cualquier acción de una forma rápida y eficaz. Es vital que no haya tiempos de espera a la hora de responder a una petición de usuario ya que esto puede confundirlo haciendo que pulse diferentes botones continuamente esperando respuesta. De este modo, se estarán mandando nuevas peticiones a la aplicación llegando a saturarla.

Claro está que para que pueda ser un e-Reader de referencia y poder competir con otros productos ya comercializados, la respuesta debe ser perfecta.

Aunque la aplicación deberá responder lo más rápido posible, en caso de que haya acciones que necesiten de más tiempo para ejecutarse, se deberá hacer ver al usuario que la aplicación sigue funcionando y que no se ha quedado colgada. Para ello, se utilizarán los mecanismos pertinentes.

- ***Fiabilidad***

El e-Reader no debe cerrarse ni colgarse al realizar ninguna de sus acciones. En caso de que eso ocurra, debe ser capaz de responder con las herramientas adecuadas para llevar a cabo la petición, o por lo menos, dejar la aplicación en el último estado en el que se encontraba. En el peor caso, deberá ser capaz de reiniciarla sin que el usuario tenga que intervenir.

2.3 Fases, duración y organización interna

Una vez se han analizado todos los requisitos y se han obtenidos los dispositivos que nos hacen falta, se podría avanzar a la fase de diseño, pero antes se van a definir cuáles van a ser las fases a seguir a partir de ahora, el tiempo que requerirán y la forma de trabajo que se seguirá.

En primer lugar, habrá que ver cómo acoplar los distintos módulos de la botonera así como hacer un estudio detallado de cómo funciona cada parte independientemente para poder hacer su posterior acople y envío de mensajes entre unos y otros. Este proceso tendrá una duración aproximada de cuatro semanas, una por cada módulo más la realización de la soldadura de los elementos que forman parte de ellos.

Una vez creado el prototipo, se pasará a la parte de programación. Habrá dos partes claramente diferenciadas: aquella que corresponde a la programación de la placa de Arduino para que envíe los mensajes correspondientes al sistema y por otro lado, la creación de una clase que permita la realizar la conexión bluetooth. Esta fase tendrá una duración superior a la anterior ya que se necesitan ciertos conocimientos, tanto de Android como del propio lenguaje de Arduino, para llevarla a cabo. Por ello, antes de empezar con el desarrollo se hará un pequeño estudio de los mismos. A esta etapa de investigación se le dedicarán tres semanas y otro mes y medio para su correspondiente implementación. Una vez la programación esté desarrollada, habrá que realizar las pruebas correspondientes para ver que el sistema funciona de acuerdo con lo establecido. Para esto se necesitará una semana y luego al menos otra más para realizar los ajustes y correcciones que sean necesarias.

La última fase será la de mantenimiento, que en principio no tiene una duración limitada ya que, como he comentado anteriormente, se orienta a todo el ciclo de vida del producto ofreciéndole al usuario un mantenimiento y actualización constante frente a posibles cambios. Por ello, habrá que preparar el sistema de versionado que será la parte más importante del mantenimiento. Se le dedicarán aproximadamente dos semanas.

Además de todo esto, cabe destacar que se ha dado la posibilidad de que un estudiante de la *Universidad Europea de Madrid* entre a formar parte del proyecto añadiendo nueva funcionalidad a la aplicación. Por ello, habrá que preparar toda la documentación necesaria para mostrársela, preparar la reunión en la que se le plantearán los requisitos y objetivos a alcanzar y hacer un pequeño estudio del estado del arte de dicha parte a desarrollar. Esta fase tendrá una duración, aproximada, de dos semanas.

Finalmente, se ha diseñado una nueva interfaz para la aplicación ya creada. Este nuevo diseño cambia totalmente la interfaz que ya estaba implementada por lo que harán falta otras tres semanas para acoplar la nueva interfaz a lo que ya teníamos previamente.

2.4 Metodología seguida

Una vez definidos todos los requisitos, adquiridos los diferentes módulos y diseñado la botonera, se comenzó a estudiar cómo funcionaban.

Inicialmente se empezó haciendo un estudio detallado de Android ya que antes de empezar con el proyecto no tenía ningún conocimiento de este lenguaje. Por ello, y sabiendo que había que desarrollar una clase que permitiese realizar la conexión bluetooth, empecé a profundizar en Android y en cómo desarrollar la clase que se quería implementar. En ese instante del proyecto se estaba trabajando con un módulo bluetooth de tipo SPP. Este tipo de perfil puerto serie está basado en la especificación 07.10 de ETSI realizando la conexión por medio del protocolo RFCOMM. Por tanto, la clase que se implementó se basaba también en este tipo de conexión.

Paralelamente se empezó a trabajar con el módulo bluetooth analizando cómo mandar los mensajes desde la placa hasta la Tablet. Lo primero que se pensó fue mandar los mensajes como cadena de texto plana y mediante diferentes hilos que se lanzaban en la aplicación, recoger dichos mensajes y tratarlos. Esto se hacía mediante un socket entre el módulo bluetooth y la Tablet pero rápidamente se vio que este método de tratamiento de los mensajes no era muy viable ya que el procesador de la Tablet no soportaba que estuviesen corriendo tantos hilos a la vez además de que el tratamiento de los mensajes en la aplicación era bastante complejo. La idea era crear una tubería desde la placa de Arduino hasta la aplicación para que esta recibiese los mensajes y los convirtiese a eventos del sistema pero se vio que el tratamiento de los mensajes debía hacerse en cada una de las clases que estaban desarrolladas y que el sistema no soportaba dicha carga. Por ello, se realizó una reunión con diferentes miembros de la empresa para pensar en diferentes alternativas.

Tras esa reunión, se sacaron diferentes ideas entre las cuales se encontraba la posibilidad de migrar a perfiles bluetooth de tipo HID. Estos perfiles, tal y como se ha comentado anteriormente, permiten mandar directamente eventos del sistema desde el módulo bluetooth y que de este modo, toda la programación recayese sobre la placa de Arduino y que la aplicación quedase en segundo plano, siendo Android el encargado de manejar dichas acciones. Además, de esa reunión también salió la posibilidad de dejar de lado la placa de evaluación y migrar la aplicación a Tablets.

Decidido esto, el siguiente paso fue profundizar en dichos perfiles HID. Se vio que para que funcionase sobre Android la versión del mismo tenía que ser superior a la 3.1 y actualmente se estaba trabajando sobre la 2.3. Por ello, antes de continuar con el desarrollo se tuvo que hacer una actualización a una versión superior y se decidió actualizar el sistema a la 4.0.3. Una vez actualizado el sistema, se estudió como se realizaba el envío de mensajes entre la placa de Arduino y el módulo bluetooth.

Para detallar cuales iban a ser los mensajes a enviar, se decidió el diseño de la botonera, cómo iban a estar colocados los botones y el joystick y cuál iba a ser la funcionalidad de los mismos. Una vez determinada la funcionalidad de los botones, se realizó un análisis de los mensajes que se debían mandar tal y como se muestra en la Tabla 1 expuesta más adelante. Cuando ya se tuvieron definidos los mensajes a mandar y se sabía el código de cada uno, se desarrolló la programación de la placa.

En ese momento, se realizaron las pruebas necesarias para comprobar el correcto funcionamiento del sistema. Se comprobó que realmente lo programado respondía a las acciones que se pretendían y que además eran las acciones que se habían determinado en la fase de requisitos. El resultado fue más que satisfactorio ya que se pudo comprobar que, además de hacer las acciones que se habían detallado, se había conseguido que el dispositivo fuese genérico, es decir, que su funcionalidad no fuese concreta de la aplicación que se había desarrollado sino que se podía controlar cualquier aplicación una vez se realizaba la conexión bluetooth entre los dos dispositivos.

Llegados a este punto el objetivo del proyecto ya se había alcanzado pero se consideró que, ahora que se estaba trabajando con Tablets y por tanto se soportaban gráficos en color, la interfaz era muy básica y se podía realizar un diseño más amigable. Entonces, se hizo una reunión con el diseñador gráfico de la empresa y se hizo un nuevo diseño de todas las interfaces de la aplicación. Una vez se tuvo el diseño, se pasó a su implementación obteniendo el resultado que se observa en el capítulo *Nuevo diseño de la interfaz de la aplicación*.

Paralelamente a todo el trabajo explicado, surgió la posibilidad de trabajar en colaboración con la ***Universidad Europea de Madrid***. A nuestro proyecto se iba a incorporar un chico y había que decidir qué parte se iba a encargar de desarrollar él. Se decidió que iba a desarrollar la lectura de libros en formato .pdf pero antes de llegar a esta decisión se tuvieron que realizar diversos análisis de los trabajos pendientes que habían quedado. Una vez tomada la decisión, se hizo la correspondiente documentación técnica para situar al nuevo miembro dentro del proyecto así como crear un control de versiones para la correcta evolución del trabajo realizado. Una vez se tuvo todo preparado, se realizó una reunión con el nuevo miembro explicándole el estado actual del proyecto y el camino a seguir.

Capítulo 3

Diseño

3.1 Estructura del sistema

3.1.1 Módulo periférico

Se va a comenzar haciendo un pequeño esquema de cómo fluirá la información entre los diferentes módulos de la *botonera* así como con la propia aplicación.

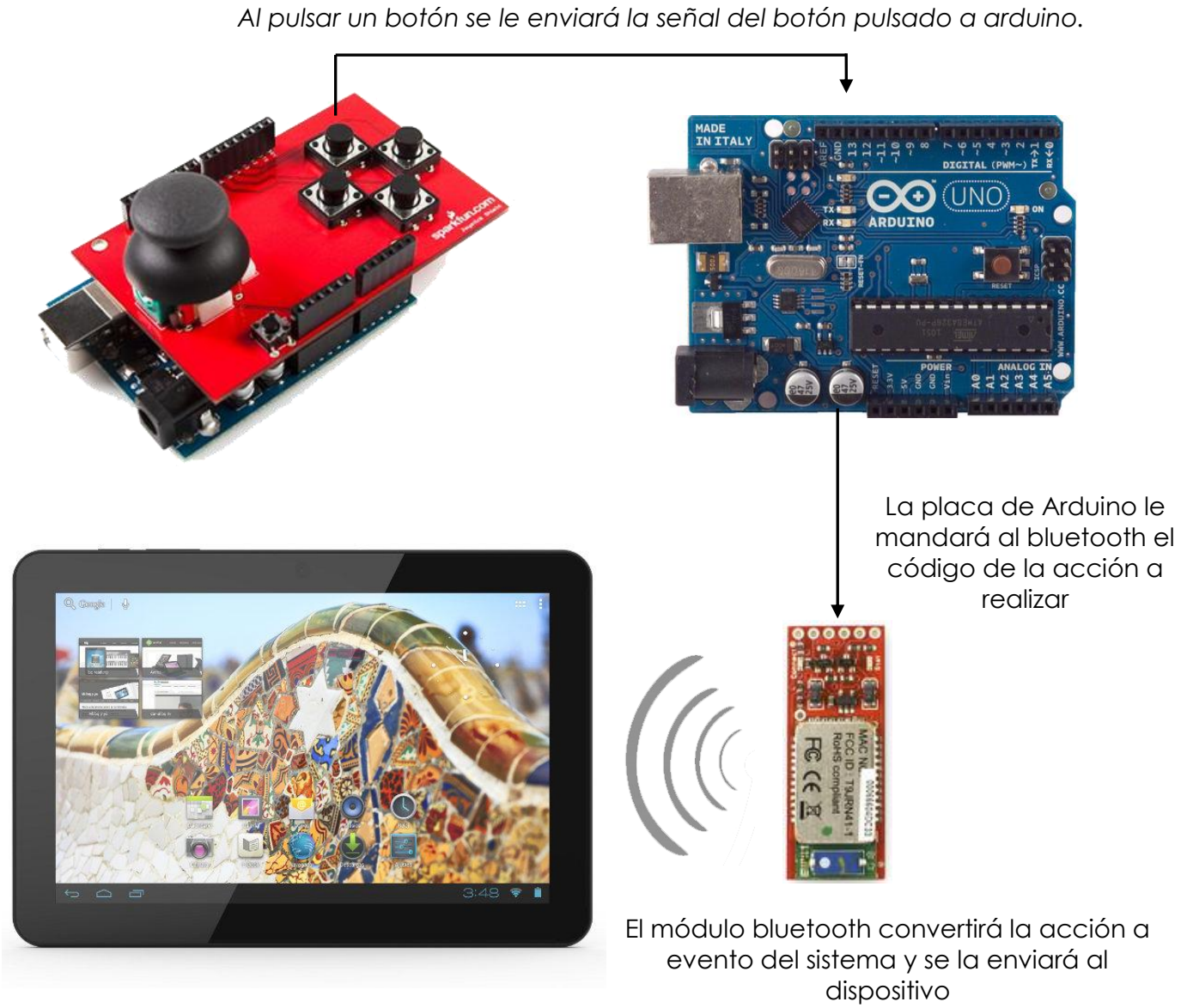


Figura 8: Esquema del flujo de mensajes entre los diferentes módulos

Tal y como se puede ver en la figura 8, al pulsar un botón de la botonera la placa de Arduino percibirá un cambio de voltaje sobre el pin al que está soldado el botón y lo asociará con un mensaje a mandar. Este mensaje se escribirá sobre el módulo bluetooth que lo recogerá e interpretará como una acción. Una vez identificada la acción a realizar, mediante el correspondiente protocolo bluetooth característico de los perfiles HID, le enviará la acción al dispositivo al cual esté vinculado. El dispositivo, al recibir el mensaje, lo tratará como evento del sistema realizando la acción que se pretendía al pulsar el botón en la botonera.

3.1.2 Nueva funcionalidad añadida

Como ya se ha comentado anteriormente, se va a desarrollar una nueva clase que permite realizar la conexión bluetooth entre el dispositivo y la botonera externa. En el siguiente diagrama vemos cómo se integra esa nueva clase en la estructura que ya teníamos desarrollada.

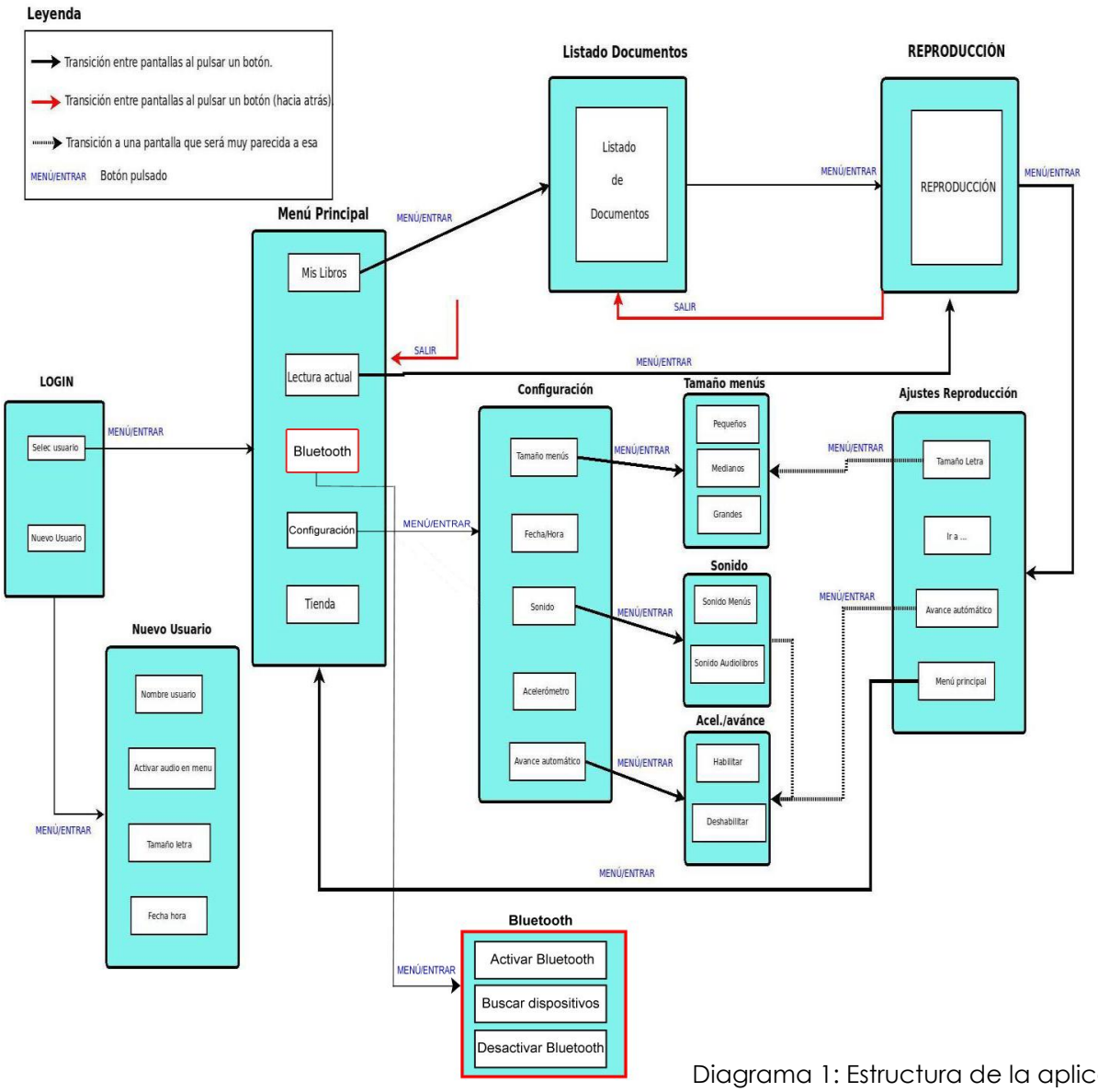


Diagrama 1: Estructura de la aplicación

Podemos ver que se trata de una clase independiente por lo que no habrá que modificar dependencias con otras clases. Sin embargo, sí que tendremos que instanciarla desde la clase principal.

3.2 Estructura de datos

Al introducir esta nueva clase (la llamada *Bluetooth* que está enmarcada dentro de un recuadro rojo) la base de datos quedará intacta ya que no afecta a la estructura de datos. Por ello, no habrá que realizar ninguna modificación sobre la misma.

3.3 Diseño funcional

3.3.1 Módulo periférico

En este apartado, se definirá un poco más detalladamente cómo fluirá la información de un módulo a otro.

Tal y como se puede ver en la figura 8, al pulsar uno de los botones de la *botonera* esta le mandará una señal analógica mediante el correspondiente pin a la placa de Arduino. Una vez tratada esa señal en la placa, se buscará el mensaje correspondiente a dicha señal y se le mandará al módulo de bluetooth. Este módulo, como ya he comentado anteriormente, al estar cargado con el perfil HID lo que hace es convertir los mensajes que le llegan en eventos propios del sistema. De este modo, la placa de Arduino le mandará el código correspondiente a la acción que queremos que realice el botón que se ha pulsado en la botonera y el módulo bluetooth, al identificar dicho código con un evento del sistema, lo transformará y lo enviará al dispositivo al cual esté vinculado. El propio dispositivo, una vez le llegue el evento, ejecutará directamente la acción a realizar.

3.3.2 Nueva funcionalidad añadida

En cuanto a la nueva clase creada, su funcionalidad será la siguiente. Al acceder a ella mediante el botón *Bluetooth* que se encuentra en la pantalla principal, el sistema comprobará si el bluetooth está disponible. En caso de que esté encendido, dará la opción de buscar dispositivos que estén dentro del alcance para así poder realizar la conexión. Si por el contrario el bluetooth está apagado, mediante el botón *Activar bluetooth* podremos activarlo y posteriormente pasar a la búsqueda de dispositivos disponibles.

Además de esto, también existe la posibilidad de desactivar el bluetooth en caso de que no se quiera utilizar la botonera. Esta acción se realizará mediante el botón *Desactivar Bluetooth*.

En ambos casos (bien si nos encontramos en el interfaz de buscar dispositivos o en el de activar/desactivar bluetooth), para volver a la pantalla anterior bastará con pulsar el botón de atrás del dispositivo que estemos manejando. Es decir, si nos encontramos en la pantalla de activar/desactivar bluetooth volveremos a la pantalla principal mientras que si lo que se está haciendo es buscar dispositivos que estén dentro del alcance, se volverá a la pantalla de activar/desactivar bluetooth.

3.4 Diseño de la arquitectura

Una vez tenemos definido el modo de funcionamiento de la clase que se va a añadir, pasaremos a ver cómo integrarla en el diagrama de clases para así ver cómo implementarla.

Podemos ver que se trata de una clase independiente a la estructura ya creada por lo que no habrá que modificar dependencias hacia otras clases. Por ello, la especificación de esta clase quedará del siguiente modo:

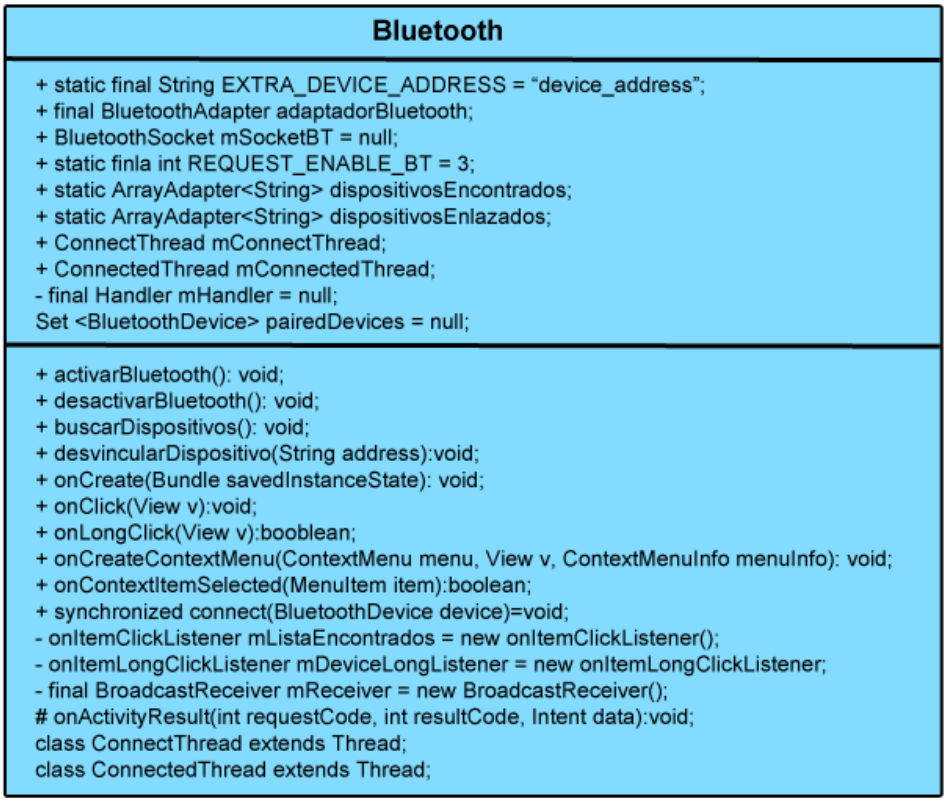


Figura 9: Especificación de la clase *Bluetooth*

Sin embargo, sí que vemos que en el menú principal hay un botón que llama a esta clase por lo que habrá que instanciarla desde la clase principal. En la figura 10 podemos ver cómo quedaría:

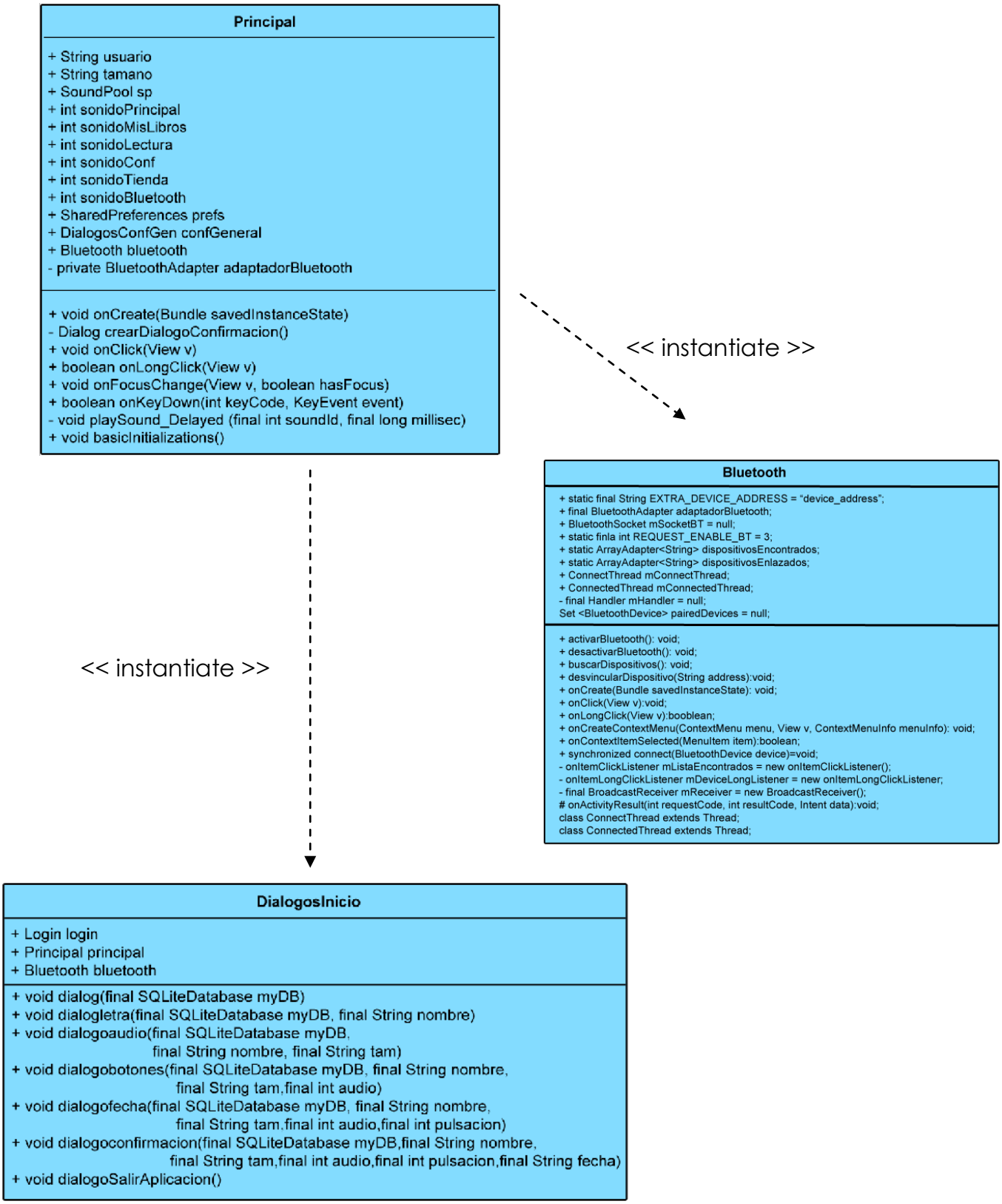


Figura 10: Diagrama de clases

Capítulo 4

Implementación

La fase de implementación la vamos a separar en tres partes: una parte se centrará en explicar la programación de la placa de Arduino para el envío de mensajes, otra parte la creación de la clase que realiza la conexión bluetooth entre el dispositivo periférico y la tablet y por último, se mostrará la migración de la vieja interfaz al nuevo diseño.

4.1 Desarrollo

4.1.1 Programación del módulo periférico

Como ya se comentó a la hora de mostrar las especificaciones de la placa de Arduino, el lenguaje de programación a utilizar es específico de la placa. El entorno de desarrollo también viene definido por defecto, tratándose de un entorno muy sencillo de utilizar.

Empezaremos explicando brevemente cómo fluyen los mensajes desde la placa Arduino hasta el dispositivo que tenemos enlazado.

Partimos del hecho de que nosotros lo que estamos haciendo es simular un teclado inalámbrico por lo que los mensajes que se envíen serán comandos de dicho teclado. El perfil HID de bluetooth interpreta los input mediante los puertos UART y por cada botón pulsado genera un evento HID que es enviado mediante la conexión bluetooth que esté establecida al dispositivo en cuestión. Estos mensajes siguen el siguiente patrón:

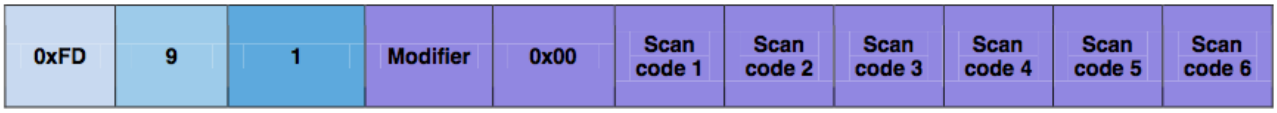


Figura 11: Patrón de mensaje HID

El primer byte **0xFD** indica que se trata de un mensaje de tipo HID. El siguiente byte indica la longitud de la cadena que estamos mandando, en este caso el número de teclas que se han pulsado. Nótese que por cada tecla que pulsemos le vamos a mandar dos señales al bluetooth, una indicándole que la tecla se ha pulsado y la otra indicándole que dicha tecla ha dejado de presionarse. Por ello, tendremos que mandarle dos mensajes. A continuación se le indica el tipo de descriptor que se está utilizando. Como nosotros lo estamos tratando como teclado inalámbrico, por defecto tendremos que ponerle el valor 1. Por último, se le indicarán los mensajes a mandar por los puertos UART. Por ejemplo, si queremos indicarle que se ha pulsado la tecla a mandaríamos lo siguiente:

0xFD	0x09	0x01	0x00	0x00	0x04	0x00	0x00	0x00	0x00	0x00

Tecla a pulsada

0xFD	0x09	0x01	0x00	0x00	0x00	0x00	0x00	0x00	0x00	0x00

Soltar tecla

Sin embargo, existe una versión reducida de estos mensajes a enviar (específica para los teclados inalámbricos) que será con lo que nosotros trabajemos. La ventaja de este otro tipo de mensajes es que se pueden enviar por los puertos UART los códigos de varias teclas al mismo tiempo con el mínimo número de caracteres posibles, minimizando de este modo el ancho de banda ya que el módulo no necesita indicar que se está trabajando como un teclado. El formato de los mensajes es el siguiente:

0xFE	Length	Modifier	Scan Code 1	Scan Code 2	Scan Code 3	Scan Code 4	Scan Code 5	Scan Code 6

Figura 12: Patrón de mensaje HID reducido

En este caso, la cabecera empieza por **0xFE** y de este modo ya sabemos que estamos tratando con un teclado y que además el mensaje que nos va a llegar es reducido. El siguiente parámetro indica la longitud del mensaje. A diferencia del patrón anterior, en este caso podemos enviar el código de más de una tecla a la vez por lo que al pulsar un botón le indicaremos que la longitud del mensaje es dos ya que uno de los códigos indicará la acción a realizar y el otro que la tecla ha dejado de presionarse.

Por ejemplo, si pulsamos las teclas a, b y c, con el formato de la figura 10, tendríamos el siguiente mensaje *a* mandar (además de indicarle que la tecla ha dejado de pulsarse):

0xFD	0x09	0x01	0x00	0x00	0x04	0x05	0x06	0x00	0x00	0x00

Sin embargo, con este nuevo formato (figura 11), el tamaño se reduce considerablemente quedando del siguiente modo:

0xFE	0x04	0x00	0x04	0x05	0x06

A continuación se va a explicar detalladamente el código desarrollado:

```
const byte BOTON_DERECHA = 3;
const byte BOTON_ARRIBA = 4;
const byte BOTON_IZQUIERDA = 6;

const byte JOYSTICK_X = 0;
const byte JOYSTICK_Y = 1;

const int X_UMBRAL_LOW = 400;
const int X_UMBRAL_HIGH = 560;

const int Y_UMBRAL_LOW = 420;
const int Y_UMBRAL_HIGH = 600;

int x_position;
int y_position;

int x_direction = 0;
int y_direction = 0;
```

En esta primera parte se encuentra la declaración de las variables. En las tres primeras declaraciones se están determinando qué pines van a utilizar cada uno de los botones que tenemos, es decir, a qué pines vamos a hacer referencia cuando pulsemos cada uno de los botones de la botonera.

A continuación, se definen los pines de las direcciones del joyStick, 0 y 1 para los ejes de coordenadas X e Y.

Después de determinar el eje de coordenadas del joyStick, se le asignan unos umbrales entres los que se limitarán los movimientos que se tomarán como válidos.

Una vez se tengan los pines que se van a utilizar declararos, así como los movimientos permitidos del joyStick, pasamos a declarar los mensajes que se enviarán al bluetooth cuando uno de los botones sea pulsado o cuando se realice algún movimiento con el joyStick.

Empezamos por los movimientos permitidos del joyStick. Al mover el controlador hacia la derecha se le enviará el mensaje indicándole que se ha pulsado la tecla de navegación de la derecha, lo mismo ocurrirá con los movimientos de izquierda, arriba o abajo. Los códigos que se le envíen tendrán el formato previamente indicado, pero en vez de mandarlos en hexadecimal, tendremos que realizar su conversión a decimal. Como ya se ha comentado más arriba, los bluetooth con perfil HID gestionan mensajes con este formato y cuando le llegue uno de los que nosotros hemos definido, directamente sabrá a qué acción nos estamos refiriendo y la convertirá a evento del sistema de Android para posteriormente enviarlo mediante el protocolo de bluetooth que esté definido.

En la tabla que se muestra a continuación podemos ver el mapeo de los códigos de las teclas que nosotros queremos gestionar en la aplicación:

Pin correspondiente al botón pulsado	*Acción a realizar*	*Código hexadecimal*	*Código decimal*
Pin 3	Enter	28	40
Pin 4	Menú	65	101
Pin 5	Salir	29	41
JoyStick derecha	Navegación derecha	4F	79
JoyStick izquierda	Navegación izquierda	50	80
JoyStick abajo	Navegación abajo	51	81
JoyStick arriba	Navegación arriba	52	82

Tabla 1: mapeo de los códigos de las teclas a simular

Por tanto, sabiendo que estos son los códigos correspondientes a las teclas que nosotros queremos simular, y tras realizar la conversión de 0xFE a decimal, los mensajes a enviar quedarán del siguiente modo:

{código teclado, longitud del mensaje, modificador, mensajes a enviar}

Tecla Enter: {254, 2, 0, 40}
Tecla Menú: {254, 2, 0, 101}
Tecla Salir: {254, 2, 0, 41}

JoyStick derecha: {254, 2, 0, 79}
JoyStick izquierda: {254, 2, 0, 80}
JoyStick abajo: {254, 2, 0, 81}
JoyStick arriba: {254, 2, 0, 82}

En la imagen inferior vemos cómo quedan dichos mensajes en el código de Arduino. Crearemos un array de cuatro posiciones por cada uno de los mensaje a enviar excepto para la tecla de release, que con un array de dos posiciones es suficiente.

```
uint8_t JOYSTICK_DERECHA[4] = {254,2,0,79};
uint8_t JOYSTICK_IZQUIERDA[4] = {254,2,0,80};
uint8_t JOYSTICK_ABAJO[4] = {254,2,0,81};
uint8_t JOYSTICK_ARRIBA[4] = {254,2,0,82};

uint8_t BOTON_ENTER[4]={254,2,0,40};
uint8_t BOTON_SALIR[4]={254,2,0,41};
uint8_t BOTON_MENU[4]={254,2,0,101};

uint8_t KEY_RELEASE[2]={254,0};
```

Después de declarar las variables del programa, tenemos que indicar el ratio de baudios que soporta la placa de Arduino así como el módulo bluetooth. En este caso son 115200bps.

```
void setup() {

  Serial.begin(115200);
  delay(320);

  pinMode(BOTON_ARRIBA, INPUT);
  digitalWrite(BOTON_ARRIBA, HIGH);

  pinMode(BOTON_DERECHA, INPUT);
  digitalWrite(BOTON_DERECHA, HIGH);

  pinMode(BOTON_IZQUIERDA, INPUT);
  digitalWrite(BOTON_IZQUIERDA, HIGH);

}
```

A continuación, declaramos los pines correspondientes a los botones de *Enter, Menú* y *Salir*. Para ello se utiliza el comando *pinMode* que lo que permite es configurar el pin especificado para comportarse como una entrada o una salida. En este caso, le indicamos que será de entrada mediante el parámetro *input*. Con la siguiente función, *digitalWrite*, se activarán las salidas digitales.

Una vez tenemos todas las variables que necesitamos declaradas, pasamos al cuerpo del algoritmo.

```
void loop() {

  x_position = analogRead(JOYSTICK_X);
  y_position = analogRead(JOYSTICK_Y);
  x_direction = 0;
  y_direction = 0;

  if (x_position > X_UMBRAL_HIGH) {
    x_direction = 1;
  } else if (x_position < X_UMBRAL_LOW) {
    x_direction = -1;
  }

  if (y_position > Y_UMBRAL_HIGH) {
    y_direction = 1;
  } else if (y_position < Y_UMBRAL_LOW) {
    y_direction = -1;
  }

  if (x_direction == -1) {
      if (y_direction == 0) {
        delay(200);
        Serial.write(JOYSTICK_IZQUIERDA,4);
      }
  } else if (x_direction == 0) {
      if (y_direction == -1) {
        delay(200);
        Serial.write(JOYSTICK_ABAJO,4);

      } else if (y_direction == 0) {

      } else {
              delay(200);
        Serial.write(JOYSTICK_ARRIBA,4);

      }
```

La función *analogRead* lee el valor de la tensión en el pin analógico especificado. Es decir, recoge el valor de la dirección que ha tomado el joystick. La placa de Arduino posee seis canales conectados a un conversor analógico digital de 10bits. Esto significa que cada vez que le llegue el valor de un pin (1 o 0 en nuestro caso), convertirá dicha tensión a un número entero de entre 0 y 1023. Con ese valor que nos devuelva, lo compararemos con los umbrales que hemos definido anteriormente y sabremos qué dirección está tomando el joystick y cuál es la acción que se debe realizar.

Empezamos por los movimientos horizontales. Si el valor de la posición x es menor que el umbral que hemos declarado como menor, se le asignará el valor -1 a dicha variable lo que indicará que, si el valor de la variable posición *y* está a cero, el joystick se ha movido hacia la izquierda. Si por el contrario la comparación con el umbral mínimo indica que es menor, teniendo la misma condición anterior, el joystick se moverá hacia la derecha. Se seguirá el mismo procedimiento para los

```
  } else {
      if (y_direction == 0) {
              delay(200);
        Serial.write(JOYSTICK_DERECHA,4);

      }
  }

  if (digitalRead(BOTON_IZQUIERDA)==LOW) {
        delay(100);
              Serial.write(BOTON_SALIR,4);
    };

     if (digitalRead(BOTON_DERECHA)==LOW) {
        delay(100);
              Serial.write(BOTON_ENTER,4);
    };

     if (digitalRead(BOTON_ARRIBA)==LOW) {
        delay(100);
              Serial.write(BOTON_MENU,4);
    };

    Serial.write(KEY_RELEASE,2);
}
```

movimientos de arriba y abajo. Cada vez que se vaya a realizar una escritura, es decir, cada vez que se vayan a escribir los mensajes en el puerto serie se detendrá el programa 200 microsegundos para evitar posibles fallos.

Para finalizar, cada vez que se pulse uno de los botones, se leerá el valor del pin digital especificado. Se realiza mediante la función *digitalRead* y dicha función devuelve HIGH o LOW dependiendo de los voltios que presente el pin en el momento en el que se presione. Además, por cada uno de los mensajes que se manden indicando que una tecla se ha pulsado, también habrá que mandarle el correspondiente mensaje indicando que la tecla ha dejado de pulsarse.

Una vez los mensajes se envíen con la función *write,* el módulo bluetooth se encargará de enviarlo mediante el correspondiente protocolo al dispositivo con el que esté enlazado.

4.1.2 Especificación de la nueva clase Bluetooth

Al inicio, cuando se estaba utilizando el módulo bluetooth con perfil SPP, se consideró la posibilidad de crear una nueva clase que permitiese al usuario establecer la conexión entre la botonera y la Tablet, ya que la interfaz que realiza dicha función no es muy accesible al público con el que nosotros estamos tratando. Por ello, se decidió crear una clase que comprobase si el bluetooth estaba activado o no y darle al usuario la posibilidad de activarlo mediante un botón y una vez activado, buscar los dispositivos que estuviesen a su alcance para poder realizar la conexión.

A continuación se van a explicar aquellos apartados que se consideran más importantes dentro del código desarrollado.

Al crear la actividad, se comprueba si el bluetooth del dispositivo está activado o no. Por defecto el botón de D*esactivar* estará desactivado y el de *Activar* visible, ya que suponemos que el bluetooth del dispositivo va a estar apagado. En caso de que esté activado, se mostrará el botón *Buscar dispositivos* y se activará el botón *Desactivar.*

```
//Comprobar si el bluetooth está activado o no y entonces activar un botón u otro y cambiar texto
 if (adaptadorBluetooth == null){
        Toast.makeText(this, "Bluetooth no disponible", Toast.LENGTH_LONG).show();
        finish();
        return;
 } else if (adaptadorBluetooth.isEnabled()){
    String txtBotonAct = btnAct.getText().toString();
    if (txtBotonAct.equals("Activar")){
        tv.setText("El Bluetooth está activado");
        btnAct.setText("Buscar dispositivos");
        btnDes.setEnabled(true);
    }
 } else {
     tv.setText("El Bluetooth está desactivado. ¿Desea activarlo?");
     btnAct.setText("Activar");
     btnDes.setEnabled(false);
 }
```

Figura 13: Segmento extraído del método *onCreate de la clase Bluetooth*

Al pulsar uno de los botones que esté activado, se realizará una u otra acción. En caso de que se pulse el botón de *Activar*, se llamará a la función *activarBluetooth*. Si por el contrario se pulsa el botón *Buscar dispositivos*, se lanzará el método *buscarDispositivos* y en caso de que se pulse el botón *Desactivar,* se pasará a ejecutar la función *desactivarBluetooh* que lo que hará será apagarlo. Todas estas funciones se explicarán más adelante. A continuación se muestra el código correspondiente (figura 13).

```
public void onClick(View v) {
    // TODO Auto-generated method stub
    switch(v.getId()){
    case R.id.btnActBlue:
        Button btnAct = (Button) findViewById(R.id.btnActBlue);
        String txtBotonAct = btnAct.getText().toString();
        if (txtBotonAct.equals("Activar")){
            activarBluetooth();
        } else {
            Toast.makeText(getApplicationContext(), "Buscando dispositivos ...", Toast.LENGTH_LONG).show();
            buscarDispositivos();
        }
        break;

    case R.id.btnDesBlue:
        desactivarBluetooth();
        break;

    case R.id.btnBuscar:
        Toast.makeText(getApplicationContext(), "Buscando dispositivos ...", Toast.LENGTH_LONG).show();
        buscarDispositivos();
        break;
    }
}
```

Figura 14: Método *onClick* de la clase *Bluetooth*

Se le ha dado la opción al usuario de elegir el tipo de pulsación que quiere realizar (si larga o corta), el código arriba mostrado se refiere a la pulsación corta de los botones pero lo mismo ocurriría en caso de que la pulsación configurada fuese larga (funcionalidad implementada en el método *onLongClick*).

Cuando se pulsa el botón *Activar* se llama al adaptador de bluetooth y se realiza la petición de encenderlo. Si el dispositivo dispone de bluetooth, se activará y mostrará un mensaje de que el bluetooth se ha activado. En caso de que el dispositivo no disponga de módulo bluetooth o no se haya activado, mostrará un error.

```
// Método para activar el bluetooth
public void activarBluetooth(){
    Intent enableIntent = new Intent(BluetoothAdapter.ACTION_REQUEST_ENABLE);
    startActivityForResult(enableIntent, REQUEST_ENABLE_BT);
}

protected void onActivityResult (int requestCode, int resultCode, Intent data){
    if(resultCode == RESULT_OK){
        ((Button)findViewById(R.id.btnActBlue)).setText("Buscar dispositivos");
        ((Button)findViewById(R.id.btnDesBlue)).setEnabled(true);
        ((TextView)findViewById(R.id.txtEstadoBlue)).setText("El Bluetooth está activado");
        Toast.makeText(getApplicationContext(), "Bluetooth Activado", Toast.LENGTH_SHORT).show();
    } else {
        Toast.makeText(getApplicationContext(),"No se ha activado el bluetooth", Toast.LENGTH_SHORT).show();
    }
}
```

Figura 15: Método *activarBluetooth* de la clase *Bluetooth*

En caso de que el bluetooth esté activado y se quiera desactivar, se llamará a la función *desactivarBluetooth* tal y como se ha comentado más arriba. Podemos ver el código desarrollado en la imagen inmediatamente inferior (figura 15).

```
// Método para desactivar el bluetooth
public void desactivarBluetooth(){
    adaptadorBluetooth.disable();
    Toast.makeText(getApplicationContext(), "Bluetooth desactivado", Toast.LENGTH_LONG).show();
    ((Button)findViewById(R.id.btnActBlue)).setText("Activar");
    ((Button)findViewById(R.id.btnDesBlue)).setEnabled(false);
    ((TextView)findViewById(R.id.txtEstadoBlue)).setText("El Bluetooth está desactivado. ¿Desea activarlo?");
}
```

Figura 16: Método *desactivarBluetooth* de la clase *Bluetooth*

En este caso, se llamará al adaptador de bluetooth y se deshabilitará. Se mostrará un mensaje indicando que el bluetooth ha sido deshabilitado y se habilitará el botón *Activar* ocultando el botón de *Buscar dispositivos* y deshabilitando el de *Desconectar.*

A continuación se pasará a explicar el código de la función *buscarDispositivos.*

```
// Método para buscar dispositivos
public void buscarDispositivos(){
    setContentView(R.layout.lista_dispositivos);
    Button btnBuscar = (Button) findViewById(R.id.btnBuscar);
    btnBuscar.setOnClickListener(this);

    // Hacemos visible el txt de los nuevos dispositivos

    // si ya está buscando se detiene
    if (adaptadorBluetooth.isDiscovering()) {
        adaptadorBluetooth.cancelDiscovery();
    }

    // Mandamos buscar
    adaptadorBluetooth.startDiscovery();

    // Inicializamos el array que va a contener los dispositivos que vaya encontrando
    dispositivosEncontrados = new ArrayAdapter<String>(this, R.layout.estilo_lista);
    dispositivosEnlazados = new ArrayAdapter<String>(this,R.layout.estilo_lista);

    // Mostrar los dispositivos que ya están enlazados
    ListView listaDispositivosEnlazados = (ListView) findViewById(R.id.lstDispEnla);
    listaDispositivosEnlazados.setAdapter(dispositivosEnlazados);
    listaDispositivosEnlazados.setOnItemLongClickListener(mDeviceLongClickListener);
    listaDispositivosEnlazados.setOnItemClickListener(mListaEncontrados);


    // mostrar los dispositivos encontrados
    ListView listaDispositivos = (ListView) findViewById(R.id.lstDisp);
    listaDispositivos.setAdapter(dispositivosEncontrados);
    listaDispositivos.setOnItemClickListener(mListaEncontrados);

    // Guardamos el registro cuando se encuentra un nuevo dispositivo
    IntentFilter filter = new IntentFilter(BluetoothDevice.ACTION_FOUND);
    this.registerReceiver(mReceiver, filter);

    // Register for broadcasts when discovery has finished
    filter = new IntentFilter(BluetoothAdapter.ACTION_DISCOVERY_FINISHED);
    this.registerReceiver(mReceiver, filter);

  // Creamos un conjunto con los dispositivos vinculados
    pairedDevices = adaptadorBluetooth.getBondedDevices();

    // Si encuentra dispositivos vinculados, los añade a la lista
    if (pairedDevices.size() > 0) {
        findViewById(R.id.txtDispEnla).setVisibility(View.VISIBLE);
        for (BluetoothDevice device : pairedDevices) {
            dispositivosEnlazados.add(device.getName() + "\n" + device.getAddress());
        }


    } else {
        //Añadimos los demás dispositivos que encuentre
        findViewById(R.id.txtDispNuevos).setVisibility(View.VISIBLE);
        String noDevices = getResources().getText(R.string.none_paired).toString();
        dispositivosEncontrados.add(noDevices);
    }

}
```

```
// BroadcastReceiver que recoge los nuevos dispositivos que se van encontrando
// y actualiza la vista cuando se acaba la búsqueda
private final BroadcastReceiver mReceiver = new BroadcastReceiver() {
    @Override
    public void onReceive(Context context, Intent intent) {
        String action = intent.getAction();

        // When discovery finds a device
        if (BluetoothDevice.ACTION_FOUND.equals(action)) {
            // Get the BluetoothDevice object from the Intent
            BluetoothDevice device = intent.getParcelableExtra(BluetoothDevice.EXTRA_DEVICE);
            // If it's already paired, skip it, because it's been listed already
            if (device.getBondState() != BluetoothDevice.BOND_BONDED) {
                dispositivosEncontrados.add(device.getName() + "\n" + device.getAddress());
            }
        // When discovery is finished, change the Activity title
        } else if (BluetoothAdapter.ACTION_DISCOVERY_FINISHED.equals(action)) {
            if (dispositivosEncontrados.getCount() == 0) {
                String noDevices = getResources().getText(R.string.none_found).toString();
                dispositivosEncontrados.add(noDevices);
            }
        }
    }
};
```

Figura 17: Método *buscarDispositivos* de la clase *Bluetooth*

Primero de todo se manda a buscar si hay algún dispositivo disponible dentro del alcance del adaptador de nuestra tablet. Si el adaptador de bluetooth ya se encuentra buscando dispositivos, se detiene. Una vez encontrados, se analizan y se irán almacenando en dos vectores diferentes. Se diferenciarán entre aquellos nuevos dispositivos encontrados y los que ya se encuentran vinculados a nuestro adaptador bluetooth.

Una vez creadas las listas, se mostrarán separándola en dos (dispositivos vinculados y nuevos dispositivos). En caso de que se pulse sobre alguno de los nuevos encontrados, se pasará a establecer la conexión y enlazar los dos dispositivos.

```
// Método para cuando seleccionamos un dispositivo de la lista de los encontrados
private OnItemClickListener mListaEncontrados = new OnItemClickListener() {
    public void onItemClick(AdapterView<?> av, View v, int arg2, long arg3) {

        // Cancela la búsqueda de nuevos dispositivos porque vamos a realizar la conexión a uno elegido
        adaptadorBluetooth.cancelDiscovery();

        // Guarda la dirección MAC. Serán los últimos 17 caracteres de la vista
        String info = ((TextView) v).getText().toString();
        String address = info.substring(info.length() - 17);

        // Creamos un dispositivo con la dirección que queremos conectar
        BluetoothDevice device = adaptadorBluetooth.getRemoteDevice(address);
        try {
            // Lanzamos el método para conectar
            connect(device);

        }
        catch (Exception e) {
            // TODO Auto-generated catch block
            e.printStackTrace();
        }
    }
};
```

Figura 18: Método *OnItemClickListener* de la clase *Bluetooth*

Para realizar dicha conexión se le llamará al método *connect*.

```
//Método para conectar con el dispositivo elegido
public synchronized void connect(BluetoothDevice device) {
    // Iniciamos un hilo para conectar con el dispositivo que se ha pasao como parámetro
    try {
        mConnectThread = new ConnectThread(device);
        mConnectThread.start();
    } catch (IOException e) {
        // TODO Auto-generated catch block
        e.printStackTrace();
    }

}
@TargetApi(Build.VERSION_CODES.ICE_CREAM_SANDWICH)
class ConnectThread extends Thread {
    public final BluetoothSocket mSocket=null;
    public final BluetoothDevice mDevice;
    public ConnectThread(BluetoothDevice device) throws IOException{
        mDevice = device;
        Method mDispositivoBT;
        try {
            //Le indicamos cómo queremos que realice la conexión (de modo inseguro)
            mDispositivoBT = device.getClass().getMethod("createInsecureRfcommSocket", new Class[] {int.class});
            mSocketBT = (BluetoothSocket) mDispositivoBT.invoke(device, 1);
        } catch (Exception e) {
            // TODO Auto-generated catch block
            e.printStackTrace();
        }
    }
    public void run(){
        adaptadorBluetooth.cancelDiscovery();
        try{
            //Lanzamos la conexión mediante el socket con el dispositivo BT que queremos
            mSocketBT.connect();
        }catch(Exception e){
            e.printStackTrace();
        }
    }

    public void cancel() {
        // TODO Auto-generated method stub
        try {
            mSocketBT.close();
            mConnectThread.interrupt();
        } catch (IOException e) {
            // TODO Auto-generated catch block
            e.printStackTrace();
        }
    }
}
```

Figura 19: Método *connect* de la clase *Bluetooth*

Este método, lanza un hilo por cada conexión que se quiera establecer. Al elegir de la lista de los posibles dispositivos el que queramos enlazar, se le pasará la dirección MAC a este método y se lanzará un hilo con dicha conexión. Entonces, se creará un socket entre los dos dispositivos que quedará a la escucha de posibles mensajes. En caso de que se quiera cerrar dicho socket, se le llamará a la función cancelar.

Si por el contrario, en vez de pulsar sobre los nuevos dispositivos que se han encontrado, se pulsa (de manera continua) sobre uno de los ya vinculados, aparecerá un menú contextual que le permitirá al usuario desvincular el dispositivo.

```
// Método que se lanza cuando presionamos durante un tiempo en un dispositivo vinculado
private OnItemLongClickListener mDeviceLongClickListener = new OnItemLongClickListener(){
    public boolean onItemLongClick(AdapterView<?> arg0, View v, int arg2, long arg3) {
        // TODO Auto-generated method stub
        // Se cancela la búsqueda en caso de que esté buscando nuevos dispositivos
        adaptadorBluetooth.cancelDiscovery();
        // Se lanza el menú contextual que nos permitirá desvincular el dispositivo en cuestión
        registerForContextMenu(v);
        return false;
    }
};
@Override
public void onCreateContextMenu(ContextMenu menu, View v, ContextMenuInfo menuInfo) {
    super.onCreateContextMenu(menu, v, menuInfo);

    // Se recoge la información de la vista que tenemos
    String info = ((TextView) v).getText().toString();
    // Extraemos el nombre del dispositivo para ponerlo de título del menú contextual
    String nombreDispositivo = info.substring(0, info.length()-18);
    menu.setHeaderTitle(nombreDispositivo);
    //Extraemos la dirección MAC del dispositivo que queremos desvincular
    EXTRA_DEVICE_ADDRESS = info.substring(info.length()-17);

    //Añadimos al menú la opción de desvincular
    menu.add(0, v.getId(), 0, "Desvincular dispositivo");

}

// Método de selección de entre las opciones del menú contextual (única en nuestro caso)
public boolean onContextItemSelected(MenuItem item) {
    // Se le llama al método para desvincular el dispositivo
     desvincularDispositivo(EXTRA_DEVICE_ADDRESS);
    return true;
}

// Método para desvincular el dispositivo que está enlazado
public void desvincularDispositivo(String address){
    BluetoothDevice dispositivoDesvincular = adaptadorBluetooth.getRemoteDevice(address);
    try {
        Method m = dispositivoDesvincular.getClass().getMethod("removeBond", (Class[]) null);
        m.invoke(dispositivoDesvincular, (Object[]) null);
        // Se le vuelve a llamar al método "buscarDispositivo" para actualizar la vista
        buscarDispositivos();
    } catch (Exception e) {
        e.printStackTrace();
    }
}
```

Figura 20: Método *desvincularDispositivo* de la clase *Bluetooth*

El método desvincular dispositivo (figura 19) lo que hace es, con la dirección MAC que se ha obtenido de la lista, invocar el método de Android de desvincular un dispositivo bluetooth. Una vez desvinculado, se vuelve a llamar a la función *buscarDispositivos* (figura 16) para que este dispositivo desvinculado aparezca como posible nuevo dispositivo dentro del alcance.

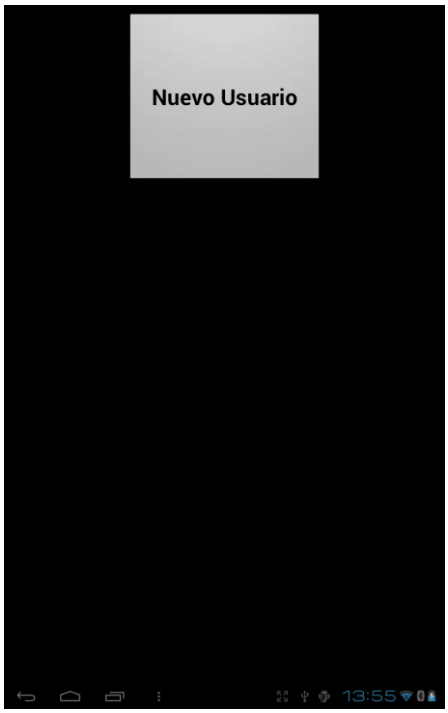
Dentro de la interfaz que muestra la lista de posibles dispositivos a conectar, se encuentra un botón que permite volver a llamar a la función *buscarDispositivos* que actualiza la lista de los mismos.

4.1.3 Nuevo diseño de la interfaz de la aplicación

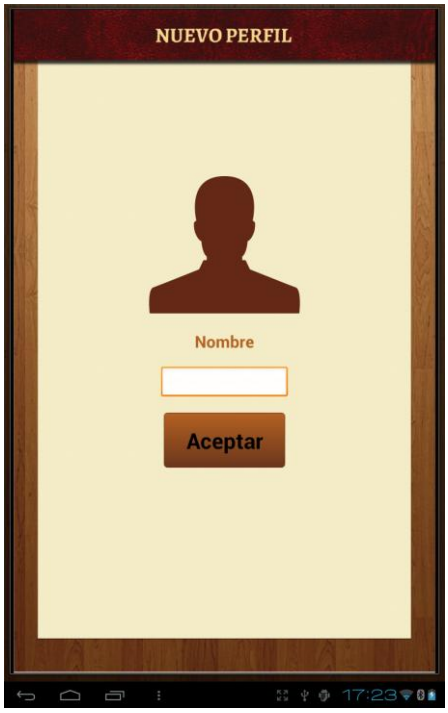
Tal y como se ha comentado a lo largo de este documento, se quiere desarrollar una aplicación que sea accesible para personas con movilidad reducida así como para aquellas personas que tengan déficit de visión. Parte importante de proyecto se centra en el diseño de la interfaz ya que se quiere conseguir una interfaz amigable, sencilla de utilizar y que sea accesible a los usuarios que hemos especificado.

Inicialmente, al empezar a trabajar con una placa de evaluación que no permitía gráficos en color, se hizo una interfaz sencilla en blanco y negro. Sin embargo, al considerar la opción de migrar la aplicación a tablets, se pensó que una interfaz en color sería mucho más amigable y se hizo un nuevo diseño de la misma. A continuación haremos un repaso de dicho cambio.

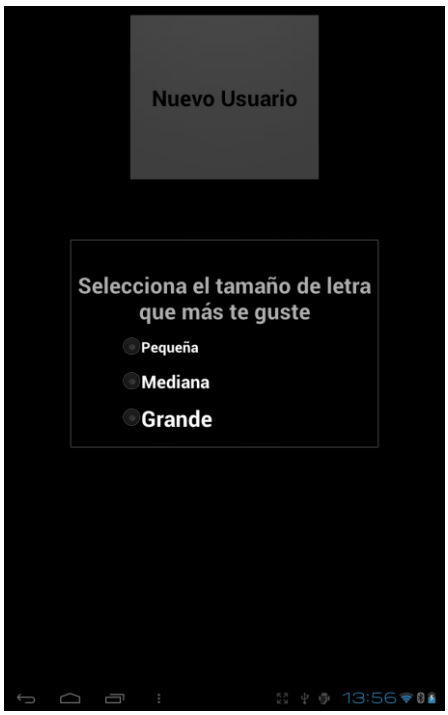
La primera vez que se lanza la aplicación no hay ningún usuario creado por lo que se tendrá que proceder a la creación de uno nuevo.

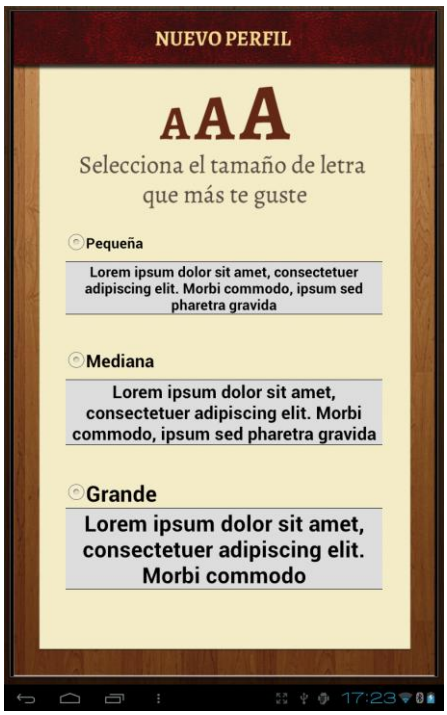


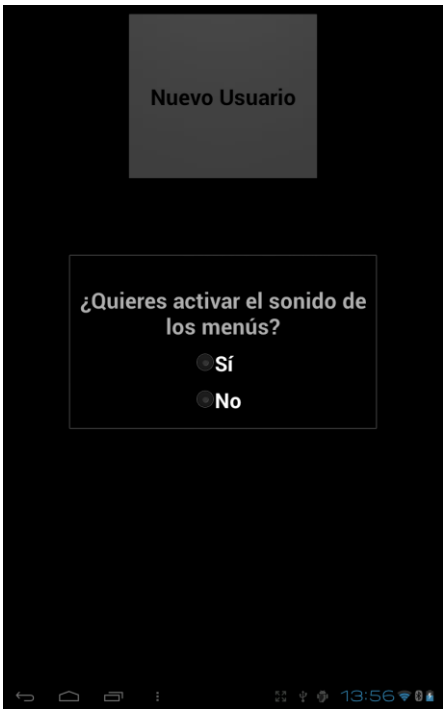
Para crear el nuevo usuario se pulsará sobre el botón *Nuevo usuario* y después de insertar el nombre que se quiera, se pasará a configurar las diferentes características que tendrá.

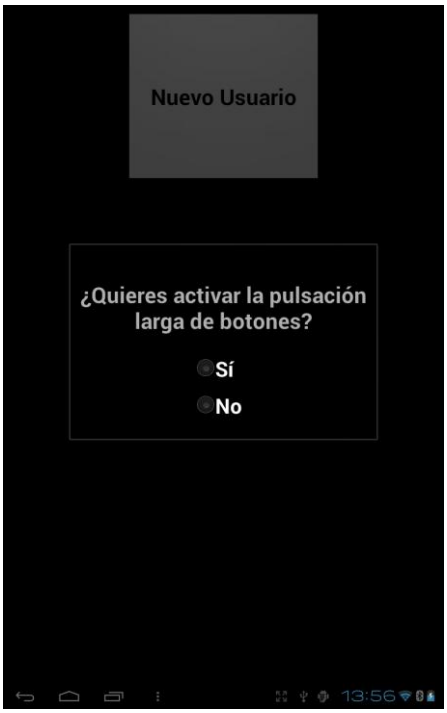


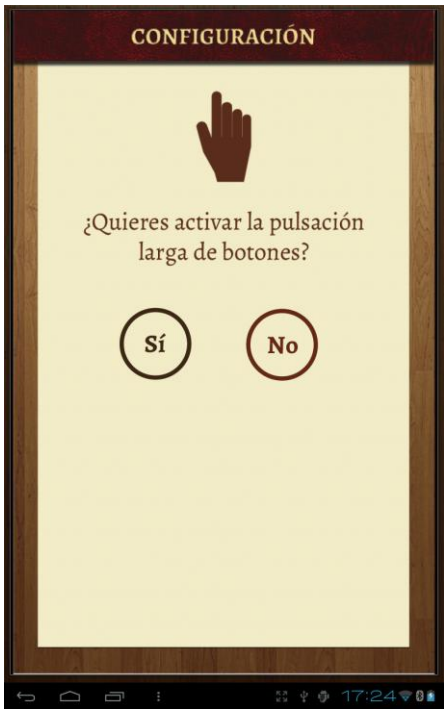
Una vez elegido el nombre identificativo, se pasa a elegir el tamaño de letra del menú, la opción de pulsación larga de los botones, sonido dentro del menú y la fecha.





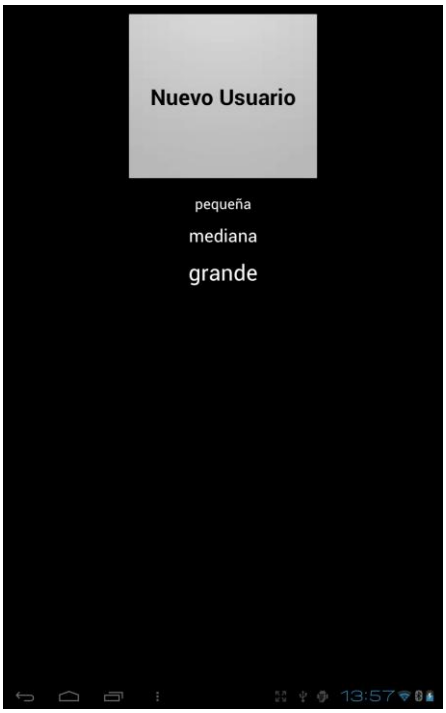






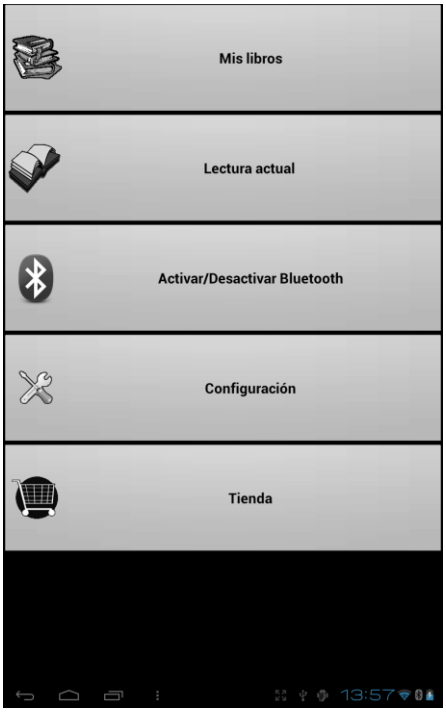
Tras elegir la configuración para el nuevo perfil que se está creando, se confirmarán las opciones elegidas.

La interfaz que se muestra a continuación será la interfaz que salga una vez se haya concluido la creación de un nuevo usuario. También será la interfaz que se cargue en caso de que ya existiese algún al iniciar la aplicación. En esta vista, aparecerán todos los posibles usuarios uno tras otro ordenados por orden de creación. Al iniciar como uno de los usuarios que aparecen, se cargarán las opciones que dicho usuario ha seleccionado así como los libros que tenga pendientes.

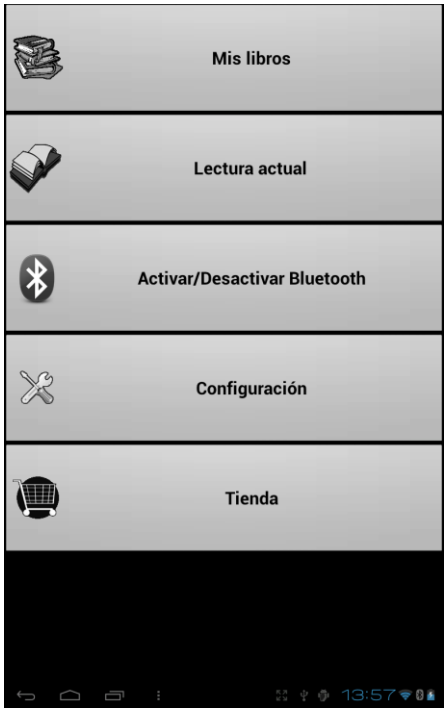


Dependiendo de las opciones que se hayan elegido a la hora de crear el usuario, el menú principal tendrá una apariencia u otra. Esto es debido al tamaño de letra que se haya elegido de entre las tres posibles alternativas (pequeña, mediana y grande).

Letra pequeña



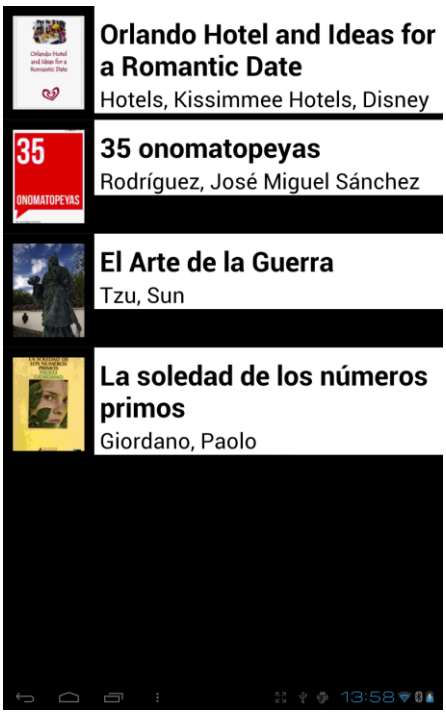
Letra mediana

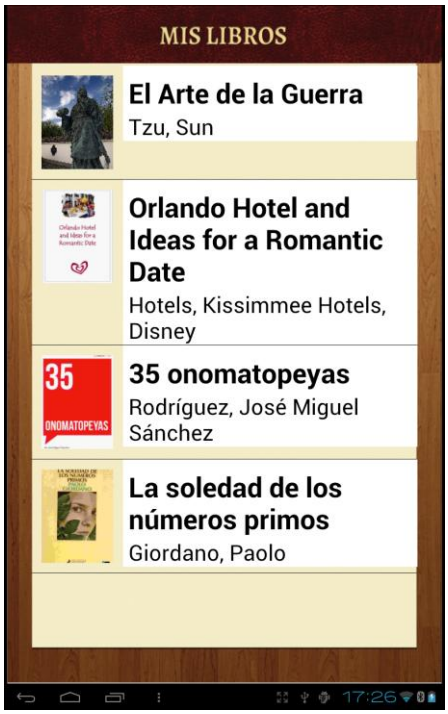


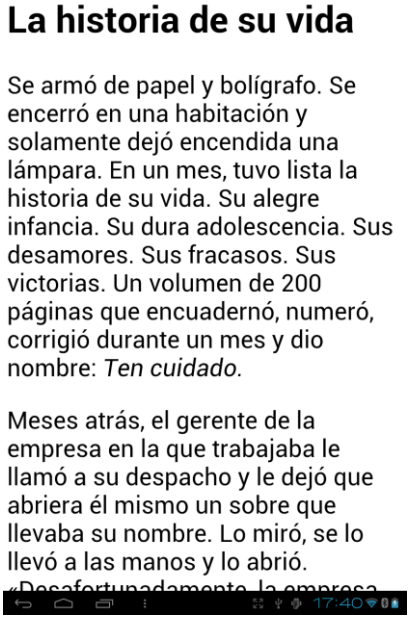
Letra Grande

Dentro del menú principal tenemos diferentes opciones entre las cuales encontramos: *Mis libros* (que permite acceder a los libros que tenemos almacenados), *Lectura actual* (vista en la que se mostrarán las lecturas que tenemos pendientes), *Bluetooth* (que dará opción de encender o apagar el bluetooth así como buscar dispositivos dentro del alcance), *Configuración* (permitirá al usuario cambiar las opciones definidas al crear el usuario) y *Tienda.*

Vamos a ir recorriendo las opciones del menú una a una. Empezamos por *Mis libros.* Al pulsar sobre dicho botón, aparecerá una lista con todos los libros que se tengan tal y como se ve en la imagen inferior.

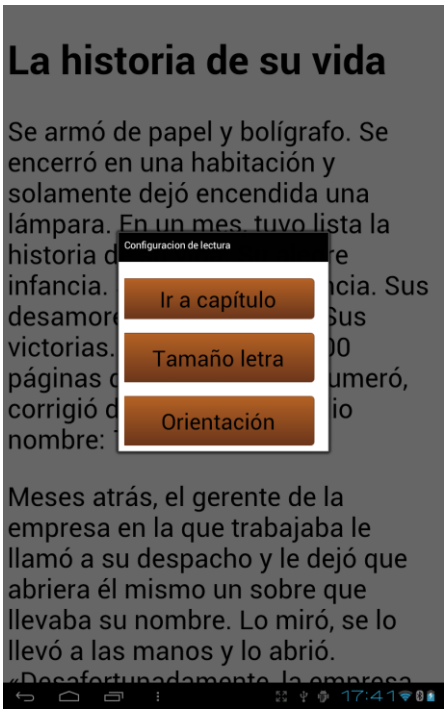




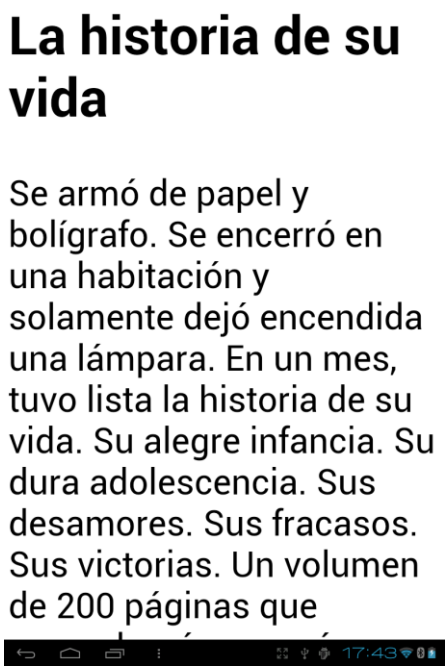


Al elegir uno de los libros que se quiera leer, se mostrará en pantalla la página primera del libro. A continuación, se podrá avanzar/retroceder página pulsando sobre los laterales de la pantalla (avanzar pulsando sobre la parte derecha y retroceder pulsando sobre la parte izquierda).

Además, dentro de esta vista, hay un menú desplegable que permite al usuario aumentar el tamaño de letra si lo desea, ir a un capítulo del libro en concreto o cambiar la orientación de la pantalla.

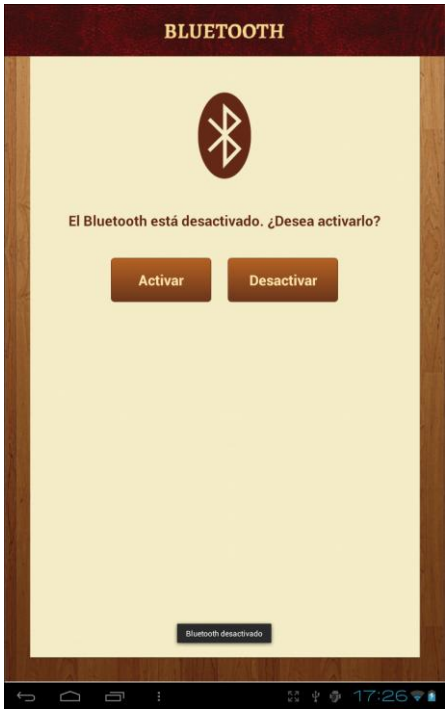


En la imagen siguiente podemos ver un pequeño cambio de letra que en este caso se ha aumentado.

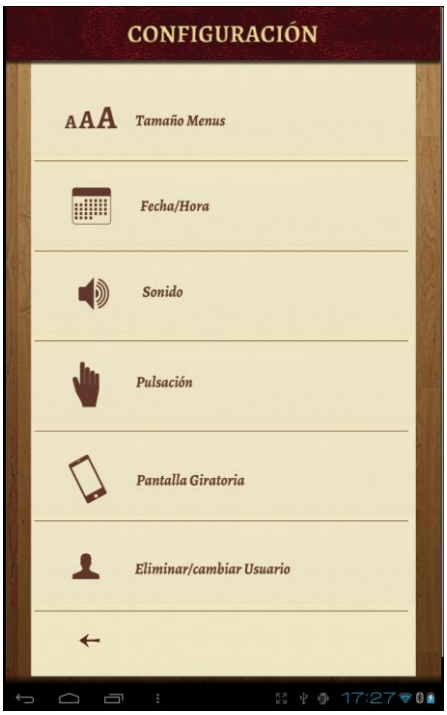


Siguiendo por el menú principal, la siguiente opción que encontramos es *Lectura Actual*. Esta vista será exactamente igual que *Mis libros* con la única diferencia que, en vez de tener la lista completa de libros disponibles, aparecerán únicamente aquellos que estén pendientes de terminar la lectura.

La siguiente opción es *Bluetooth*. Tal y como se ha comentado, esta pantalla permite al usuario encender y apagar el bluetooth así como buscar los dispositivos que estén a su alcance. Esta clase ha sido implementada en esta nueva parte del proyecto por lo que no existe interfaz anterior con la que comparar.



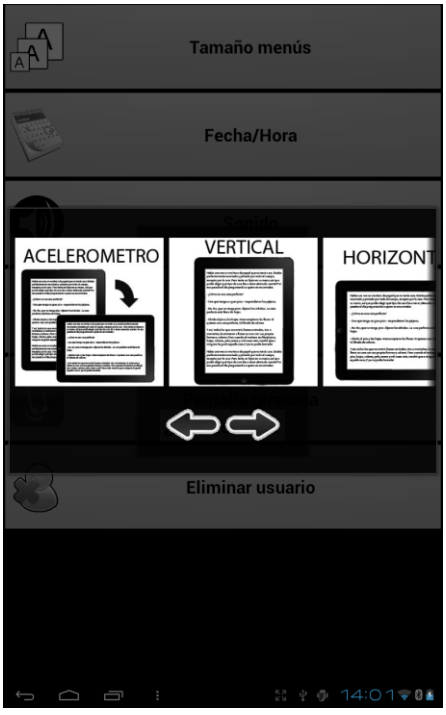
Por último, encontramos la opción de *Configuración*. Esta vista nos da la opción de cambiar aquellos parámetros que se han definido en la creación del usuario. Al igual que sucedía con el menú principal, el tamaño de la letra de las opciones de configuración también variará dependiendo de lo que se haya seleccionado inicialmente. A continuación vemos cómo quedaría la interfaz si se selecciona el tamaño de letra *mediano*.



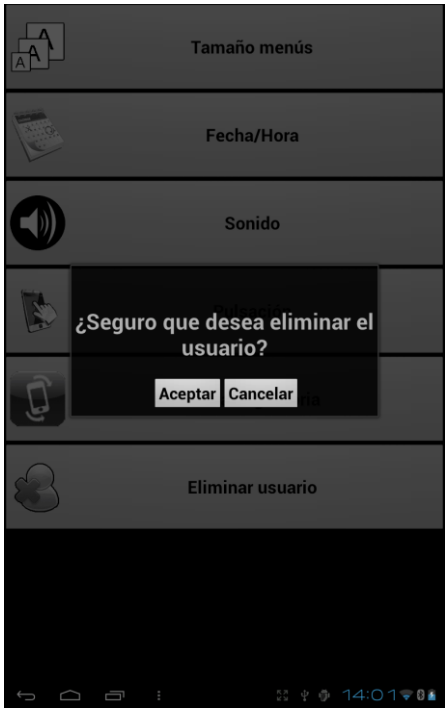
Dentro de estas opciones, la mayoría de pantallas de configuración de los diferentes parámetros son iguales a las que se han utilizado para la creación de nuevos usuarios ya que las características a configurar son las mismas. Por ello, nos vamos a centrar únicamente en aquellas que no han sido explicadas todavía.

Ajustes de *Fecha y Hora*.

Ajustes de *Pantalla giratoria*.



Y por último, la opción de *Eliminar usuario*.



La última opción del menú principal es *Tienda* pero se trata de un apartado que todavía no ha sido desarrollado y se ha dejado pendiente como trabajo futuro, por lo que no tiene interfaz implementada.

4.2 Pruebas

Las pruebas son una parte fundamental del proyecto y deben planificarse ya que no todas se realizarán al finalizar el desarrollo de la aplicación. De esta manera, se han realizado diferente tipos de pruebas durante el desarrollo del proyecto:

Las pruebas que se han realizado para este proyecto han ido en incremento, es decir, se han realizado en primer lugar en cada uno de los módulos y a medida que se ha visto su correcto funcionamiento se ha pasado a juntar todos los módulos hasta finalmente completar todo el sistema.

Para realizar este proceso incremental, en primer lugar, se planificó la integración de los módulos ya que hay elementos que se comparten y en caso de una mala gestión, podrían afectar al correcto funcionamiento de la botonera así como de la aplicación. Para evitar este tipo de problemas se siguieron los siguientes pasos:

- Antes de realizar cualquier intento de acoplamiento, se guardará una copia de seguridad de lo que se tenga desarrollado.
- A la hora de introducir la nueva clase desarrollada, se creará una nueva clase donde se irá añadiendo el contenido de la misma en las diferentes clases desde las que se le haga la llamada. De esta manera, se facilitará el proceso de unificación.

El propósito de las pruebas de unidad es probar los componentes implementados como unidades individuales. Se han realizado dos tipos de pruebas:

- **Pruebas de especificación o caja negra,** que verifican el comportamiento de la unidad observable externamente. Se han realizado para verificar el componente siguiendo unas determinadas entradas que producen unas determinadas salidas. Esto se comprobó en la botonera. Se comprobó que al pulsar uno de los botones de la misma se realizaba la correspondiente acción sobre la aplicación. Es decir, si se pulsaba el botón Intro había que comprobar que efectivamente se activaba dicho botón en la aplicación navegando a la siguiente pantalla. Lo mismo ocurría con el joystick, cuando se le indicaba que navegase hacia abajo, había que comprobar que el cursor realizaba lo correspondiente, moviéndose hacia abajo.

  En definitiva, se puso a prueba el sistema y se observó que cumplía con los requisitos especificados y que la aplicación respondía a acciones que supuestamente se le estaban mandando pero en todo momento, sin fijarnos en qué pasaba por dentro, sólo centrándonos en los resultados obtenidos.

- **Pruebas de estructura o caja blanca,** que verifican la implementación interna de la unidad. Para cada componente se ha estudiado su implementación interna y se ha tratado de verificar su correcto comportamiento algorítmico. Se comprobaron los umbrales del joystick viendo que al realizar ciertos movimientos fuera de esos umbrales especificados, el cursor no realizaba la función que se le había mandado previamente. Además, también hubo que modificar varios códigos de los mensajes que se enviaban ya que con alguno de los mensajes (más concretamente con el mensaje de salir/atrás) las acciones que se realizaban no eran las correctas.

  Es decir, en este caso lo que se ha hecho ha sido intentar emular todas las posibles situaciones por las que puede pasar la aplicación para ver que realiza las acciones correctamente.

Tal y como se ha comentado, estas pruebas se realizaron en primera instancia aplicándolas sólo a los diferentes módulos y cuando ya se tuvo constancia del correcto funcionamiento de cada uno de ellos, se realizaron las mismas pruebas pero para el conjunto de la aplicación. De esta manera, se consiguió acotar los errores más fácilmente.

4.3 Mantenimiento

Al tratarse de un producto que va a ser lanzado al mercado, es muy importante incorporar las herramientas suficientes para poder hacer que el sistema funcione correctamente durante todo el ciclo de vida. Es decir, puede pasar (y es lo normal que así ocurra) que una vez esté comercializado, se empiece a vender y cuando los usuarios lo utilicen falle en algún momento pese a que se hayan realizado pruebas para comprobar su correcto funcionamiento.

Además, se está trabajando sobre un sistema operativo que está en continua evolución por lo que habrá que ir desarrollando nuevas adaptaciones a dichos cambios para dar soporte a nuevas versiones de Android.

Por otro lado, puede pasar que con el paso del tiempo el usuario requiera nuevas funcionalidades y por tanto habrá que implementarlas.

Para resolver ésta problemática, se definirá un plan de mantenimiento.

El objetivo de esta fase es definir cómo se va a mantener el sistema en funcionamiento mientras esté siendo utilizado. La solución recae en ir creando versiones del sistema que soluciones los posibles errores o que añadan nueva funcionalidad. Con esta metodología se abarcan los dos problemas planteados ya que al instalar una nueva versión, se solucionan los problemas que han sido identificados y se añaden las nuevas funcionalidades que se hayan desarrollado.

En todo momento se ha querido facilitar el uso de la aplicación al usuario por lo que en este caso la idea de actualizar a una nueva versión también debe ser sencilla. El usuario deberá conectarse a internet desde el dispositivo que esté utilizando (bien sea vía wifi o 3G) y al conectarse, el propio dispositivo identificará que existe una nueva versión de la aplicación y le preguntará al usuario si desea actualizar. En caso de que el usuario acepte el proceso, este será automático y el usuario no tendrá que intervenir en ningún momento.

Capítulo 5

Líneas futuras

En lo que a las ***líneas futuras*** respecta, se pueden diferenciar dos caminos por los que seguir trabajando:

Dispositivo periférico y módulos añadidos

- *Nuevos dispositivos periféricos.* Una vez observado el correcto funcionamiento de la botonera desarrollada, se pasará a diseñar nuevos módulos externos que permitan el control de la aplicación. Estos nuevos dispositivos serán sopladores o pedales.
- *Diseñar prototipos de acuerdo a las necesidades del cliente*. Inicialmente se hizo un diseño de lo que sería el prototipo de la botonera pero se debería hacer un estudio con gente con diferentes discapacidades para diseñar diferentes prototipos que abarquen todos los usuarios potenciales.
- *Realizar pruebas con diferentes usuarios.* Actualmente se han realizado pruebas con usuarios con déficit visual y con cierta movilidad reducida pero sería interesante realizar pruebas con usuarios que tengan otro tipo de discapacidades.

Aplicación de e-Reader

- *Añadir más formatos de lectura*. Actualmente se está trabajando en la opción de leer archivos de tipo .pdf.
- Introducir audiolibros.
- *Finalizar el proceso de grabación de los sonidos del menú*. la funcionalidad está desarrollada en varias pantallas pero no se han utilizado sonidos definitivos sino demos de prueba. Por tanto, se deberá decidir cómo se quieren, grabarlos y añadirlos a la aplicación.
- *Desarrollar la tienda online*. habrá que preparar tanto la aplicación como el sistema para que permita conectarse a internet sin que se produzcan errores.
- *Preparar manejo de versiones*. Tal y como se ha comentado en la fase de mantenimiento, para hacer que el producto no quede desfasado y

para solucionar posibles errores, habrá que preparar versiones que modifiquen el sistema. Por ello, por un lado habrá que trabajar en cómo crear las nuevas versiones y por otro lado, ver cómo hacer para que el sistema adopte los nuevos cambios.

- *Realizar más pruebas con usuarios potenciales.* Hasta el momento se han realizado pruebas con usuarios con discapacidad visual.
- *Desarrollar o implementar un sistema de reconocimiento de audio.* Sería interesante, sobre todo para usuarios que tienen algún tipo de discapacidad física, que se implemente un sistema de reconocimiento de audio o de patrones, para poder usar el e-Reader mediante señales de voz.

Capítulo 6

Conclusiones

Técnicamente se puede decir que el objetivo de este proyecto está más que alcanzado. El diseño de la botonera creada no está limitado a un grupo de usuarios concreto ya que, al ser tres módulos independientes, la placa de Arduino se puede adaptar a cualquier otro tipo de dispositivo, bien sea soplador, pedal o cualquier combinación que el usuario precise. Es decir, se puede hacer un diseño detallado centrado en las necesidades de cada usuario realizando cambios mínimos en la programación de la placa base.

Además, al haber utilizado un tipo de módulo bluetooh con perfil HID, la señales que se mandan a través de la conexión bluetooth no son tratadas específicamente en la aplicación desarrollada sino que cualquier aplicación externa a nuestro proyecto puede ser manejada perfectamente.

Es por ello por lo que podemos decir que finalmente, se ha conseguido desarrollar un producto que va más allá de la idea inicial y que su evolución se considera un éxito.

Hasta que no llegué a Job Accommodation no era consciente de lo que la palabra *accesibilidad* significaba. La RAE lo define como "*Calidad de ser de fácil acceso*" pero también se entiende como *la relación con las tres formas básicas de la actividad humana: movilidad, comunicación y comprensión, sujetas a limitación como consecuencia de la existencia de barreras*. Y en estas barreras son en las que nos hemos centrado en este proyecto fin de carrera. Barreras que, a simple vista, son invisibles a nuestros ojos pero que son el día a día de un extenso grupo de esta sociedad.

Lo que hace dos años pretendía ser un simple dispositivo periférico que permitiese el manejo de una aplicación de lectura electrónica accesible, se ha convertido en un dispositivo genérico que permitirá (y permite) la utilización, de la forma más autónoma y natural posible, de dispositivos electrónicos. De este modo, se ha querido dotar a la sociedad de una herramienta que haga que cualquier persona pueda disponer y utilizar servicios o productos en igualdad de condiciones que los demás, proporcionando flexibilidad al acomodarse a las necesidades de cada usuario.

Se dice que mientras más avance la tecnología, más complicada será nuestra existencia. Y este es el momento de cambiar el significado de esta cita. O incluso borrarla y volver a escribirla. En nuestras manos está el poder de decidir cuán complicados queremos que sean los avances tecnológicos y cuán accesible queremos que sea nuestra aportación a la sociedad, pues nosotros somos esa generación que puede decidir qué hacer y cómo hacerlo, tal y como se lanzaron allí por el 2009 Ion Esandi y Patxi Fabo, CEOs de Job Accommodation. Llegará un día en el que todos precisemos de tecnología accesible, bien porque se tenga algún tipo de discapacidad o simplemente por el paso de los años, y es por ello que me alegra poder decir que he podido formar parte de un proyecto empresarial que no es más que la punta del iceberg de lo que será algún día.

En definitiva, durante el transcurso de este proyecto he aprendido cómo funciona una empresa desde dentro, he tenido la posibilidad de experimentar cómo gestionar un proyecto cuando hay miembros que se unen a él una vez comenzado su desarrollo y, quizá lo más importante, ha hecho cambiar mi forma de ver la tecnología y su usabilidad.

Bibliografía

Referencias bibliográfica

1. Fernando Alonso, Loïc Martinez., Fco. Javier Segovia (2005). *Introducción a la Ingenieria del Software: Modelos de desarrollo de programas,* Delta Publicaciones.
2. Joan Ribas Lequerica (2013). *Desarrollo de aplicaciones para ANDROID.* Anaya Multimedia.
3. Rick Rogers, John Lombardo, Zigurd Mednieks (2009). *Android Application Development: Programming with the Google SDK.* O'Reilly.
4. Michael Margolis (2011). *Arduino Cookbook.* O'Reilly.

Referencias en línea

1. **Android**
   - 1.1. http://developer.android.com/about/index.html
   - 1.2. http://developer.android.com/training/index.html
   - 1.3. http://developer.android.com/guide/components/index.html
   - 1.4. http://developer.android.com/reference/packages.html
   - 1.5. http://developer.android.com/sdk/index.html
   - 1.6. *Plugin de eclipse*
     - 1.6.1. http://developer.android.com/sdk/installing/installing-adt.html
2. **Arduino**
   - 2.1. http://arduino.cc/es
   - 2.2. *Software*
     - 2.2.1. http://arduino.cc/es/Main/Software
   - 2.3. *API*
     - 2.3.1. http://arduino.cc/es/Reference/HomePage
3. **zTerm**
   - 3.1. www.dalverson.com/zterm
4. **Bluetooth**
   - 4.1. **BlueSMiRF Gold** https://www.sparkfun.com/products/10268
   - 4.2. **BlueSMiRF HID** https://www.sparkfun.com/products/10938

Agradecimientos

Todo este cambio en mi forma de pensar y de ver las cosas no hubiese sido posible de no haberse presentado delante de mí la posibilidad de realizar el proyecto en Job Accommodation y de no haber podido trabajar y conocer a todos los compañeros que forman parte del equipo de esta pequeña empresa.

Es por eso que lo único que queda por decir es GRACIAS, en mayúsculas. En primer lugar, a todos y cada uno de los miembros de Job Accommodation, por el trato recibido desde el minuto uno, por hacerme sentir parte de la empresa y por dejarme la oportunidad de gestionar este proyecto que tan buen resultado ha obtenido. Especial mención a Gabriela, por su paciencia y su continua sonrisa.

A Iker Marinelarena, por confiar en mí y por pensar que era la persona idónea para continuar con el proyecto que es tanto mío como suyo. Por su apoyo incondicional año tras año.

Por otro lado, agradecer a José Javier Astráin toda la ayuda ofrecida para hacer frente a los problemas que han ido surgiendo durante las diferentes fases del proyecto y por aceptar la propuesta de ser tutor de este proyecto fin de carrera.

A mis compañeros de clase, por esta última fase de nuestra vida universitaria. Por todos esos momentos vividos.

A Macarena y Zuriñe, por su confianza. Por regalarme su visión externa y demostrarme que al final, todo sale.

A mis padres y hermana, porque este proyecto es tan suyo como mío. Por ser un apoyo en los momentos malos y vivir como suyos los logros obtenidos.

Y finalmente, agradecer a toda mi familia, amigos y compañeros por estar siempre dispuestos a ayudarme en todo.

Muchas gracias. Mila esker.